University of Alberta

LOGIC-ORIENTED FUZZY MODELS AND FUZZY MODELING

by

Xiaofeng Liang

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring, 2008

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-45555-5
Our file Notre référence
ISBN: 978-0-494-45555-5

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

As the complexity of systems increases, their successful modeling becomes a difficult and complex task. Key challenges in system modeling include constructing accurate as well as transparent and highly interpretable models easily comprehended by humans. With this respect, the development of user-centric models endowed with highly interactive interfaces is a highly relevant and timely task.

The objective of our research is to investigate and develop a generalized logic model that is able to achieve a meaningful balance between accuracy and transparency when interacting with users. Such a model can deal efficiently with highly dimensional modeling problems. Fuzzy logic and fuzzy sets are able to cope with linguistic information (information granules) and are therefore compatible with human perception. We exploit the technology of fuzzy neural networks. Such networks combine the superb learning abilities of neurocomputing with the high interpretability aspects associated with fuzzy logic. The design scheme consists of three fundamental phases, namely the design of efficient information granulation mechanisms realized by the interface layout, the formation of learning schemes in the processing core, and the interpretation of model, delivering readable rules back to the user. Several design techniques are presented in the thesis including fuzzy equalization, conditional Fuzzy C-means clustering, particle swarm optimization, gradient-based learning, and network pruning. Experimental studies are reported and the obtained results demonstrate the feasibility and efficiency of the proposed models.

Acknowledgements

Foremost, I especially would like to express deepest gratitude and thanks to my supervisor, Dr. Witold Pedrycz, for his indispensable and invaluable guidance that help me to achieve my dream in the past five years. He provides support and encouragement through difficult periods of the research, bringing a spirit of exploring science to the project during my PhD studies at the University of Alberta. The completion of this thesis would be impossible without his close supervision and devotion to education. His patience, innovations, enthusiasm, and prestigious contributions in Computer Intelligence development and Software Engineering make him a great mentor.

Additionally, I want to heartily thank my wife Xiaomeng Yang, my parents, Jinshu Liang and Xiumei Li, parents-in-law, Jishun Yang and Hongying Wang, and my sister Xiaowen Liang, without their support, trust and encouragement, I could not have obtained today's achievement.

TABLE OF CONTENTS

Chapter 1 Introduction 1
1.1 Motivation 1
1.2 Objectives 2
1.3 Main contributions 3
1.4 Dissertation Organization 5
Chapter 2 Fuzzy Modeling Fundamentals 8
2.1 Neural Networks – Literature Review 8
2.2 Fuzzy Sets and Fuzzy Logic 10
2.3 Fuzzy Modeling and Neurofuzzy Modeling 12
2.4 General Architecture of the Fuzzy Model 15
2.5. Interfaces of a Fuzzy Model 16
2.6 Conclusions 18
Chapter 3 Logic-Based Neurons 25
3.1. AND and OR Logic Neurons 25
3.2 Characteristics of AND and OR Logic Neurons 27
3.2.1 Input-output characteristics of AND neurons 27
3.2.2 Input-output characteristics of OR neurons 29
3.3 Fuzzy AND and OR unineurons 32
3.4 Characteristics of AND and OR unineurons 35
3.4.1 Input-output characteristics of AND unineurons 35
3.4.2 Input-output characteristics of OR unineurons 37
3.5 Conclusions 40
Chapter 4 Architecture of Logic-Based Networks 42
4.1 The topology of AND and OR neuron-based neural networks 42
4.2 The topology of AND and OR unineuron-based neural networks 50
4.3 Interpretation of logic networks 55
4.4 The design of the network 58
4.4.1 Formation of contexts through fuzzy equalization 58
4.4.2 Conditional Fuzzy C-Means in the formation of the blueprint of the logic network 59
4.4.3 Projection and reduction of input variables 60
4.5 Conclusions 62
Chapter 5 Discretization 64
5.1. Terms and notations 64
5.2 Approaches to discretization – Literature survey 65
5.3 Proposed modeling environment 70
5.4 Particle swarm optimization (PSO) 71
5.5 Illustrative example 73
5.6 Conclusions 74
Chapter 6 Learning 78

6.1 Assessment of the overall development of a logic network 78
6.2 Internal and external performance index of the network 80
6.3 Gradient-based learning 81
6.4 Learning with the use of PSO 84
6.4.1 Strategy-1 PSO for all parameters of the network that are binary {0, 1} 85
6.4.2 Strategy-2 PSO for all parameters that are in the range of [0, 1]. 85
6.4.3 Strategy-3 Hybrid of PSO and gradient-based learning 85
6.5 Fuzzy partition 87
6.6. Conclusions 88
Chapter 7 Experimental studies 90
7.1 Networks constructed by AND and OR neurons 90
7.1.1 Boston housing data 90
7.1.2 Auto-MPG dataset 95
7.1.3 Computer dataset 97
7.1.4 Plasma Retinol Levels 99
7.1.5 Air pollution at a road -- NO2 dataset 101
7.2 Networks constructed by AND and OR unineurons 103
7.2.1 Boston Housing dataset 103
7.2.2 Auto-MPG dataset 110
7.2.3 Abalone dataset 111
7.2.4 Computer dataset 113
Chapter 8 Conclusions and Future Work 116

LIST OF FIGURES

Figure 2.1 The structure of a three-layered feed-forward neural network 9
Figure 2.2 Fuzzy model interpreted by extracted fuzzy rules 13
Figure 2.3 General topology of a logic model 16
Figure 2.4 Filter vs. wrapper strategies of feature subset selection 17
Figure 2.5 Two-phase processing flow chart of an input interface 18
Figure 3.1 Logic processing of neurons: AND neuron 25
Figure 3.2 Logic processing of neurons: OR neuron 26
Figure 3.3 Input–output characteristics of an AND neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.05 0.30]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives. 28
Figure 3.4 Input–output characteristics of an AND neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.69 0.30]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives. 29
Figure 3.5 Input–output characteristics of an OR neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.05 0.30]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives. 30
Figure 3.6 Input–output characteristics of an OR neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.65 0.23]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives 31
Figure 3.7 Computing of uninorms 33
Figure 3.8 Logic processing of unineurons (a) AND unineuron and (b) OR unineuron 34
Figure 3.9 Input–output characteristics of AND unineurons for selected pairs of t- and s-norms. In all cases, the corresponding connections and identity points are set to **w** = [0.05 0.30] and **g** = [0.40 0.60], respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives. 36
Figure 3.10 Input–output characteristics of AND unineurons for selected pairs of t- and s-norms. In all cases, the corresponding connections and identity points are set to **w** = [0.69 0.30] and **g** = [0.30 0.70], respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives. 37
Figure 3.11 Input–output characteristics of OR unineurons for selected pairs of t- and s-norms. In all cases, the corresponding connections and identity points are set to **w** = [0.05 0.30] and **g** = [0.40 0.60], respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives. 38
Figure 3.12 Input–output characteristics of an AND unineuron for selected pairs of t- and

s-norms. In all cases, the corresponding connections and identity points are set to **w** = [0.65 0.23] and **g** = [0.50 0.30], respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives. 39
Figure 4.1 A topology of the SOM type logic network 43
Figure 4.2 Characteristics of SOM network for selected values of the connections and realizations of t- and t-conorms(min and max) 44
Figure 4.3 Characteristics of SOM network for selected values of the connections and realizations of t- and t-conorms(product and probabilistic sum) 45
Figure 4.4 Characteristics of SOM network for selected values of the connections and realizations of t- and t-conorms (Lukasiewicz t and s-norm) 46
Figure 4.5 A topology of the POM type logic network 47
Figure 4.6 Characteristics of POM network for selected values of the connections and realizations of t- and t-conorms (min and max) 48
Figure 4.7 Characteristics of POM network for selected values of the connections and realizations of t- and t-conorms (product and probabilistic sum) 49
Figure 4.8 Characteristics of POM network for selected values of the connections and realizations of t- and t-conorms (Lukasiewicz t and s-norm) 50
Figure 4.9 A topology of a multiple-input-single-output SOM network 51
Figure 4.10 Characteristics of unineuron-based network for selected values of the connections and realizations of t- and t-conorms(min and max) 52
Figure 4.11 Characteristics of unineuron-based network for selected values of the connections and realizations of t- and t-conorms(product and probabilistic sum). 53
Figure 4.12 Characteristics of unineuron-based network for selected values of the connections and realizations of t- and t-conorms (Lukasiewicz t and s-norm) 54
Figure 4.13 A topology of a multiple-input-single-output POM network 55
Figure 4.14 Structure of a fuzzy model represented as an aggregation of fuzzy rules 55
Figure 4.15 Conditional Fuzzy C-Means: a flow of computing 60
Figure 4.16 A concept of the context-based clustering; note that each context induces "c" clusters in the input space 60
Figure 4.17 Projection of prototypes along with the contexts and induced clusters in the input space 61
Figure 4.18 Merging of close fuzzy sets and construct of contexts using updated fuzzy sets 62
Figure 5.1 Equal-width discretization 66
Figure 5.2 Equal-frequency discretization 66
Figure 5.3 A hierarchy of discretization methods 70
Figure 5.4 Mechanism of discretization by K-Means 70
Figure 5.5 The pseudo-code of PSO 72
Figure 6.1 The optimization of network parameters **v** and **w** 80
Figure 6.2 Particle with binary value encoded 85
Figure 6.3 Partiicle with unit interval value [0, 1] encoded 85
Figure 6.3 Hybrid learning scheme 86
Figure 6.4 The development of fuzzy sets based on use of the original Boolean partition ... 87
Figure 7.1 A histogram of MEDV 91
Figure 7.2 Fuzzy sets constructed in the output space 91
Figure 7.3 Number of fuzzy sets for each input variable with respect to different 92
Figure 7.4 Overall number of fuzzy sets for the inputs with 3,5 and 9 clusters 92
Figure 7.5 Values of the performance (training and testing set) treated 94
as a function of K and L 94
Figure 7.6 Fuzzy equalization of the MPEG output: 95

(a) histogram, and (b) resulting fuzzy sets 96
Figure 7.7 Fuzzy equalization of output PRP 97
(a) A histogram of PRP; (b) Three linguistic labels 97
Figure 7.8 Fuzzy equalization of RETPLASMA level 100
(a) histogram and (b) resulting fuzzy sets 100
Figure 7.9 Fuzzy equalization of the NO2 101
(a) the histogram and (b) resulting fuzzy sets 101
Figure 7.10 The number of neurons in the hidden layer vs. performance index Q 106
Figure 7.11 Performance index V a function of ε 110
Figure 7.12 Inconsistency rate after discretization 110
Figure 7.13 Training and testing performance index V vs. core parameter ε 111
Figure 7.14 Inconsistency rate after discretization 112
Figure 7.15 Training and testing performance index V vs. core parameter ε 113
Figure 7.16 Inconsistency rate after discretization 113
Figure 7.17 Training and testing performance index V vs. core parameter ε 114

LIST OF TABLES

Table 2.1 Plots of selected t-norms (min, product, Lukasiewicz, and drastic product) 11
Table 2.2 Plots of selected s-norms (max, probabilistic sum, Lukasiewicz, and drastic sum) 11
Table 4.1 Fuzzy equalization for triangular fuzzy sets A1, A2, …, Ap 58
Shown above are formulas for the parameters of the corresponding fuzzy sets 58
Table 5.1 Discretization results for the synthetic data 74
Table 7.1 The interpretation of the network for MEDV = L 93
Table 7.2 The interpretations of networks for MEDV = M and H 94
Table 7.3 Interpretation of the network for MPG ={S, M, L} 96
Table 7.4 The interpretations of networks for PRP = {LOW, MEDIUM, HIGH} 98
Table 7.5 The interpretations of networks for RETPLASMA ={LOW, MEDIUM, HIGH} 100
Table 7.6 Fuzzy sets formed in the input space (5 clusters per context; ε=0.7) 101
Table 7.7 The interpretation of the networks 102
Table 7.8 Inconsistency rate ζ as a function of p and c 104
Table 7.9 Comparison of inconsistency rates between two discretization methods. 104
Table 7.10 Distribution of cutoff points by PSO $\triangle$ (triangle-up) vs. Equal-width discretization ∇ (triangle-down). 104
Table 7.11 Strategy-1 PSO training Boolean parameters $\{0,1\}^p$ 106
Table 7.12 Strategy-2 PSO training continuous parameters $[0,1]^p$ 107
Table 7.13 Strategy-3 Hybrid learning (PSO and gradient-based learning) of continuous parameters $[0,1]^p$ 108
Table 7.14 AND-OR network trained by gradient-based learning of continuous parameters $[0,1]^p$ 109
Table 7.15 Comparison of training and testing performance index Q among three learning strategies 111
Table 7.16 Comparison of training and testing performance index Q among three learning strategies 112
Table 7.17 Comparison of training and testing performance index Q among three learning strategies 114

Chapter 1
Introduction

1.1 Motivation

As the complexity of systems increases, their successful modeling becomes a difficult and complex task. Neural networks (NNs) [4][7] and fuzzy logic (fuzzy sets)[17] approaches have been proven to be efficient vehicles to cope with challenging modeling issues when applied to a variety of problems emerging in numerous areas of technology and science[1-2][8-10][13-15]. There has also been a growing interest in combining the two technologies, giving rise to what is commonly referred to as neurofuzzy models and neurofuzzy modeling [3][6][18]. These hybrid architectures have started to receive more attention and thus have evolved substantially in the past decade.

However, with the rapid growth of neurofuzzy modeling, we are faced with several fundamental challenges. One of the most important is accuracy. The accuracy of a model is two-fold. First, the model must be able to capture the nature of the data by minimizing a certain performance index, such as the root mean squared errors (RMSE)[11], where the error is measured by comparing the real output to the output produced by the system. Secondly, the model must produce correct results when output is computed for new data. Another challenge is the need for high transparency in the fuzzy models. Surprisingly, with the inception of neurofuzzy systems and the growing dominance of evolutionary computing as a vehicle of global optimization, the notion of transparency of fuzzy models seems to have been somewhat overlooked—in spite the fact that readability and ease of comprehension of fuzzy models were driving forces behind the inception of the entire area several decades ago.

The architecture of fuzzy models is predominantly focused on processing information granules. The external world for which the models are formed is to a very high extent numeric. Because they are focused on handling information granules, fuzzy models are abstract constructs. In light of this, there is a need to construct efficient interfaces between the modeling environment and fuzzy models; this functional module calls for an effective way of interfacing numeric and granular information. One of the fundamental tasks of the fuzzy modeling agenda is the construction of interfaces of such nature.

Moreover, in order to enhance the model's performance, there is usually a genuine need to develop an efficient interaction with the designer of the model. In particular, it is highly beneficial to facilitate a seamless interaction between

the model designer and the identification environment. In this manner we are able to accommodate and take full advantage of domain knowledge and experience. User-centric modeling aims to create models with higher human interactivity and friendliness.

Motivated by the existing challenges, this research seeks to construct a generalized logic model that is able to achieve an optimal and effective balance between accuracy and transparency by accepting user interactive inputs. Such a model can efficiently deal with problems of many dimensions and with the involvement of human interaction.

1.2 Objectives

This study is a continuation of ongoing research on *logic-driven* fuzzy models. There are several objectives of this research:

1. Revisit and systematization of existing logic processing units with respect to their functionality, underlying logic, interpretation aspects, and learning abilities:

These logic processing units include fuzzy neurons and unineurons [5][12][16]. With respect to their functionality, underlying logic, interpretation aspects, and learning abilities, some have been well documented in the existing literature, but some still require systematization and further investigation. In this research, we aim to systematically explore the properties of these logic processing units.

2. Development of logic structures of fuzzy models based on different types of neurons:

Our ultimate goal is to develop a logic network with superb learning ability and transparent interpretability. The logic processing units discussed in our research, including fuzzy neurons and unineurons, are conceptually simple logic-oriented elements that come with well-defined semantics and plasticity. Owing to their diversity, such neurons form essential building blocks of any fuzzy model architecture. By arranging the neurons in successive layers, we can produce a rich collection of meaningful logic expressions and nonlinear characteristics of input-output mappings.

3. Enhancement of model interpretability and improvement of effective readability through pertinent pruning mechanisms:

The advantage of a network built by fuzzy neurons and unineurons resides with its significant interpretability capabilities. There are several parameters that

directly impact the readability of the results. In particular, the numeric connections of neurons determine the weights of the conditions in the rules, and the number of neurons in the hidden layers indicates the number of rules generated. The introduction of a pruning mechanism sheds light on ways to further reduce the network, in this way improving its interpretability.

4. Forming mechanisms of information granulation of experimental data:

Real-world data are continuous while logic-based processing realized by fuzzy models operates on the more abstract constructs of information granules. The phase of granulation of information, and discretization in particular, has to be carefully investigated. The impact of the discretization scheme on the quality of data formed in this manner is critical and needs to be quantified. Suitable criteria are of interest when dealing with an assessment of the level of possible distortion introduced via discretization and its impact on the performance of the model

5. Consideration of various schemes of the development of networks with special emphasis on the parametric aspects of learning and its realization in terms of evolutionary optimization, such as particle swarm optimization (PSO) and the hybrid of PSO with gradient-based learning:

The issue of optimization is important so the logic constructs have to exhibit a significant level of parametric flexibility. Optimization thus requires a suitable development environment. There are quite a few learning schemes available in the literature. Among them, evolutionary optimization and gradient-based learning are two sound options to be explored in this regard. In order to fully exploit these learning schemes, we need to compare and investigate the performance of these methods and their hybrid.

1.3 Main contributions

The findings of our research activities contribute to the field of logic-driven modeling in the following ways:

1. Construction of a logic network with the aid of fuzzy neurons and unineurons:

Motivated by the need to construct networks that exhibit plasticity and retain interpretability, we have developed two types of fuzzy modeling frameworks which have the flexibility to model a wide range of problems in various fields. The first type of framework consists of a set of logic AND and OR neurons, and the second type of framework is built by fuzzy unineurons. Because of the general nature of unineurons, the second type of framework exhibits more

flexibility than the first. Both networks offer unique synergy in transparency and learning.

2. Approaches suitable for parametric optimization:

Three learning strategies are proposed for model identification. These learning strategies are capable of discovering concise, human-interpretable logic-based structures in data, which are then further refined in order to achieve high levels of accuracy and generalization while retaining transparency.

3. Assessment of strengths and weaknesses of different discretization methodologies:

We identify strengths and weaknesses of different methodologies of discretization in the literature and assess the impact of these discretization schemes on the quality of data formed for processing. We thus propose design guidelines and a new discretization method that copes with the current limitations. Through extensive experimentation we reveal and quantify the quality of the discretized data. The fact that this study takes into consideration information granules of output space and all of the variables in the input space makes this contribution unique.

4. A complete investigation of model performance by Boolean data and induced fuzzy data:

Based on the Boolean partition created by discretization, we introduce fuzzy sets to the resulting intervals to form the induced fuzzy data. Such fuzzy data has high level flexibility and further helps investigate the performance of the logic model introduced in this thesis. We report on the performance of the logic model after a comprehensive suite of experiments and deliver several interesting conclusions for further development.

5. User interactive design procedure including a pruning mechanism:

In order to generate simple and easily understood logic expressions, user interactive designs are introduced in this thesis. The pruning mechanism, whose intention is to simplify the logic network, takes advantage of the logic nature of the neurons and removes some of the neurons' connections, thus enhancing the network's interpretability. Such a mechanism is highly user-oriented: its interpretation benefits from the reduced form of the model by the acceptance of somewhat higher values of the approximation error.

1.4 Dissertation Organization

The dissertation is organized as follows:

Chapter 2 Fuzzy modeling fundamentals
This chapter provides a comprehensive literature review of fuzzy modeling. We include a discussion on neural networks and logic operations and also discuss the interpretability of these networks. A general framework of fuzzy modeling is presented in this chapter.

Chapter 3 Basic logic-based neurons
This chapter gives a detailed introduction to the generic constructs of fuzzy neurons, including AND and OR neurons, as well as the fuzzy uninorm-based unineurons. These logic neurons are the basic processing units of the logic networks discussed in the next chapter.

Chapter 4 Architecture of logic-based neural networks
In this chapter we introduce the architecture of networks in detail. Several design issues are presented that contrast with other topologies. For example, fuzzy equalization, context-based clustering, etc. We also investigate the interpretability of these networks.

Chapter 5 Discretization
This chapter offers a comprehensive overview of the state of the art of discretization. We also propose a hybrid discretization and discuss the detailed design issues and performance evaluation criteria. Investigations are conducted to show the importance of this new development methodology.

Chapter 6 Learning strategies
In this chapter we discuss the strengths and weaknesses of several learning strategies. Based on the investigation, we then propose three learning strategies for the aforementioned fuzzy model. Detailed design issues for the learning are also covered in this chapter.

Chapter 7 Experimental studies
This chapter presents extensive experiments that demonstrate the effectiveness and transparent interpretability of the networks.

Chapter 8 Conclusions and future work
In this chapter, we draw conclusions from our work; we review our contributions to the field and consider future work in the area of system modeling and knowledge discovery.

Bibliography

[1] R. Babuska, Fuzzy Modeling for Control, Kluwer Academic Publishers, Boston, 1998.
[2] S.P. Chitra, "Using neural networks for problem solving", Chemical Engineering Progress, pp. 44-52, 1993.
[3] S.R. Gunn, M. Brown, and K. M. Bossley, "Network performance assessment for neurofuzzy data modelling", Advances in Intelligent Data Analysis: Reasoning about Data, Springer Verlag Publishers, pp. 313-324, 1997.
[4] S. Haykin, Neural Networks: a Comprehensive Foundation, Prentice Hall, New Jersey, 2nd edition, 1999.
[5] K. Hirota, and W. Pedrycz, "OR/AND neuron in modeling fuzzy set connectives", IEEE Transactions on Fuzzy Systems, vol. 2, pp. 151-161, 1994.
[6] J.S.R. Jang, and C.T. Sun, "Neurofuzzy modeling and control", IEEE Transactions on Fuzzy Systems, vol. 3, pp. 378-406, 1995.
[7] P.A. Jansson, "Neural networks: an overview", Analytical Chemistry, vol. 63, pp. 357-362, 1991.
[8] C.L. Karr, and E.J. Gentry, "Fuzzy control of pH using genetic algorithms", IEEE Transactions on Fuzzy Systems, vol. 1, pp. 46-53, 1993.
[9] D. Kim, and C. Kim, "Forecasting time series with genetic fuzzy predictor ensemble", IEEE Transactions on Fuzzy Systems, vol. 5, pp. 523-535, 1997.
[10] M.T. Leung, W.E. Engeler, and P. Frank, "Fingerprint processing using backpropagation neural networks", Proceedings of the International Joint Conference on Neural Networks I, pp. 15-20, 1990.
[11] N. Levinson, "The wiener RMS (root mean square) error criterion in filter design and prediction", Journal of Mathematics and Physics, vol. 25, pp. 261-278, 1947.
[12] W. Pedrycz, and K. Hirota, "Uninorm-based logic neurons as adaptive and interpretable processing constructs", Soft Computing, vol. 11, no. 1, pp. 41-52, 2007.
[13] D.A. Pomerleau, Neural Network Perception for Mobile Robot Guidance, Boston: Kluwer, 1993.
[14] N. Quian, and T.J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models", Journal of Molecular Biology, vol. 202, pp. 865-884, 1988.
[15] T.J. Sejnowski, B.P. Yuhas, M.H. Goldstein, and R.E. Jenkins, "Combining visual and acoustic speech signals with a neural network improves intelligibility", Advances in Neural Information Processing Systems, vol. 2, pp. 232-239, 1990.
[16] R.R. Yager, "Uninorms in fuzzy systems modeling", Fuzzy Sets and Systems, vol. 122, pp. 167-175, 2001.
[17] L.A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic", Fuzzy Sets and Systems, vol. 90, no. 2, pp. 111-127, 1997.
[18] Q.Y. Zhang, and A. Kandel, "Compensatory neurofuzzy systems with fast

learning algorithms", IEEE Transactions on Neural Networks, vol. 9, no. 1, pp. 83-105, 1998.

Chapter 2
Fuzzy Modeling Fundamentals

In order to investigate the strengths and weaknesses of the existing logic modeling techniques, in this chapter we offer a comprehensive review of the underlying concepts of neural networks, fuzzy logic, and fuzzy neural networks, starting with an overview of the general architecture of fuzzy models. We then provide detailed architectural considerations and a discussion of several key techniques involved in fuzzy modeling.

2.1 Neural Networks – Literature Review

The history of neural networks can be tracked back to the earliest simulation of a biological neuron proposed by McCulloch and Pitts in 1943 [42]. The model of the neuron was a simple linear threshold computing unit with two inputs and a single output. The output was activated only when the inputs summed and exceeded a threshold level. In the next neural model, the perceptron, developed by Rosenblatt in 1958 [74], weights were introduced to the neurons. With a change of weights, the neurons could achieve certain "learning," but the adjustment of weights was based on trial-and-error. In 1958 Selfridge suggested a process to update the weights—he named it "mountain climbing." This process is now referred to as the gradient descent method. In Selfridge's method, update of the weights was guided by a randomly assigned direction vector [78]. If the performance of the neuron did not improve, the weights were returned to their previous values and a new random direction vector was assigned. Widrow and Hoff developed another type of gradient search method for adapting the weights [87]. Their approach was based on minimizing the squared error, a method known as the least mean squares algorithm (LMS). LMS reduced the computing time, making perceptron a useful neuron model which received much attention over the following years [75].

However, in 1969 Minsky and Papert pointed out that perceptrons can solve only linearly separable problems[44]; a single perceptron is incapable of representing simple functions that are linearly inseparable such as the "exclusive or" (XOR). In order to solve an n-separable problem, "n-1" perceptrons are needed. After Minsky and Papert's book was published, the area of neurocomputing fell into a decline during the 1970s.

In 1974 Werbos described the back propagation (BP) algorithm [86], but the method received little attention. In 1985 and 1986, three researchers, Parker,

Rumelhart and McClelland, independently re-proposed the back propagation algorithm [51, 70]. The BP algorithm is supervised learning in which perceptrons are arranged in a multilayer fashion and trained in so that n-separable problems can be solved. The BP algorithm also enhanced perceptrons by replacing the threshold function by a sigmoidal function. The development of the BP algorithm led to a renaissance in neural networks in the 1980s and 1990s [4, 64, 73, 79, 89]. Since then, neural networks have established a reputation in various areas of modeling and prediction such as robot navigation [48, 49], robotic telemanipulation and robotic tactile recognition [65-68], modeling of 3D objects [10, 11, 63, 64], simulation of a dielectric ring resonator antenna [70], speech recognition [80], handwritten character recognition [37], natural gas load prediction [50], wastewater treatment plant control [30], fingerprint recognition [36], etc.

Various types of neural networks are found in the literature [19, 20, 22, 23, 31, 32]. For instance, in a feed-forward neural network, we organize neurons in a sequence of layers with the layers being fully connected. Such networks have been implemented to solve a variety of problems in various branches of engineering and science. Figure 2.1 shows a typical architecture of the feed-forward neural network.

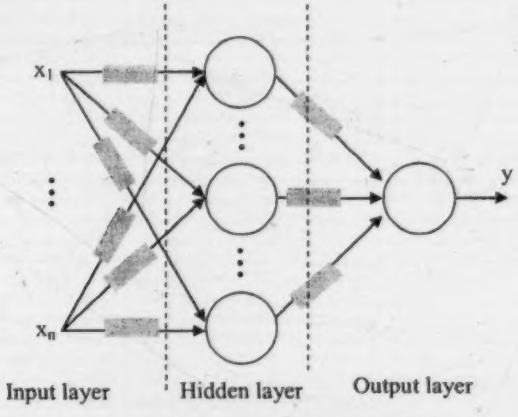


Figure 2.1 The structure of a three-layered feed-forward neural network
(the inputs are treated as a single layer)

The BP algorithm suffers from the limitation of falling into the local minima during the learning. Therefore there is interest in introducing new techniques into neural networks to alleviate this problem. Genetic Algorithms (GAs) [16, 45] have been used successfully in this context and global learning techniques have substantially improved the performance of neural networks constructed in this manner.

Another drawback to neural networks is a lack of transparency; that is, knowledge acquired in a trained network is difficult to interpret. Thus a neural network is sometimes viewed as a black-box model which realizes a mapping from $R^n \rightarrow R^m$, where n and m are the dimensions of input and output, respectively.

2.2 Fuzzy Sets and Fuzzy Logic

The principles of fuzzy sets and fuzzy logic were introduced by Lotfi Zadeh several decades ago [91]. A fuzzy set is an extension of a crisp set which allows the elements discussed to either belong to or not belong to the set; there is a clear boundary between "belong" and "not belong." Fuzzy logic is known to cope with linguistic information granules [53, 59, 91] in line with ideas of human perception and strongly supporting the development of transparent models. For instance, humans usually use the concepts "young" and "old" to describe age. If a person is younger than 40 years old, we may say he is "young." By introducing linguistic terms [53, 59, 91] we can further describe these concepts. Instead of "young" and "old" we can use "very young," "young," "old," and "very old" to describe a person's age. Such characterizations benefit from well-defined fuzzy sets.

Triangular norms (t-norm) and triangular conorms (t-conorm, or s-norm) are two fundamental logic operations in fuzzy logic [8, 14, 18, 29]. t-norm is a generalization of the Boolean logical conjunction; t-norm is a function $t: [0, 1]^2 \rightarrow [0, 1]$ which satisfies the following properties:

Commutativity: $t(a, b) = t(b, a)$
Monotonicity: $t(a, b) \leq t(c, d)$ if $a \leq c$ and $b \leq d$
Associativity: $t(a, t(b, c)) = t(t(a, b), c)$
Identity element 1: $t(a, 1) = a, \quad a \in [0,1]$

Similarly, t-conorm (s-norm) is a function $s: [0, 1]^2 \rightarrow [0, 1]$ which satisfies the properties of commutativity, monotonicity, and associativity with $s(a, 0) = a$ for all $a \in [0,1]$. Clearly, the min operator is a t-norm, and max is an s-norm, and can be regarded as generalized set intersection and union operations. The t-conorms are dual to the t-norms. Given a t-norm, the dual t-conorm is:

$$s(a,b) = 1-t(1-a, 1-b)$$

which generalizes De Morgan's laws.

In the literature there are several examples of t-norms and t-conorms. Among them, four pairs of t- and s-norms are commonly used, namely min-max, product-probabilistic sum, Lukasiewicz t-s, and drastic t-s. Table 2.1 and Table 2.2 summarize these norms by means of 3D plots and 2D contour plots.

Table 2.1 Plots of selected t-norms (min, product, Lukasiewicz, and drastic product)

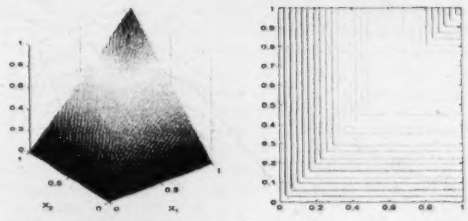
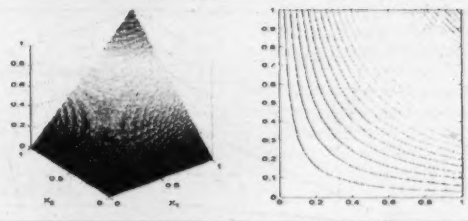
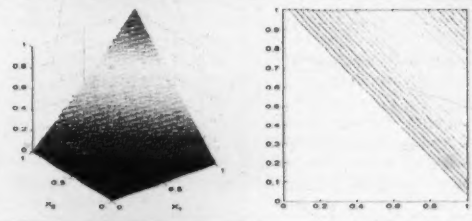
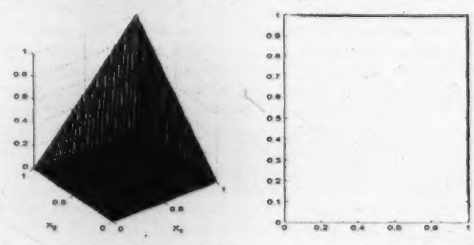
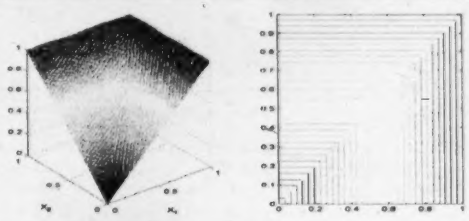
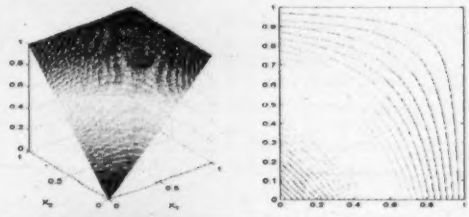
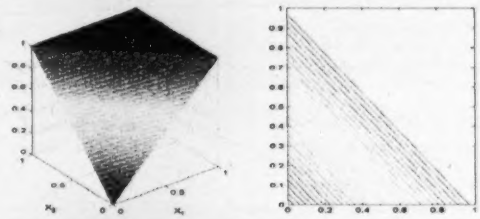
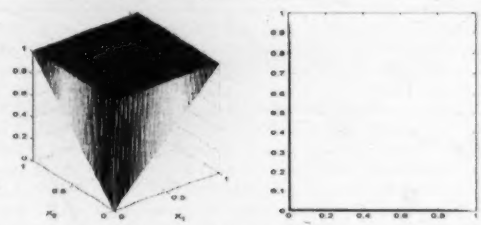
t-norm	Plots of the t-norm (3D and 2D contours)
Min $t(a,b)=\min(a,b)$	
Product $t(a,b)=a\times b$	
Lukasiewicz $t(a,b)=\max(0,a+b-1)$	
Drastic Product $t(a,b)=\begin{cases} b & a=1 \\ a & b=1 \\ 0 & \text{otherwise} \end{cases}$	

Table 2.2 Plots of selected s-norms (max, probabilistic sum, Lukasiewicz, and drastic sum)

s-norm	Graph of the s-norm (3D and 2D contours)

Max s(a,b)=max(a,b)	
Probabilistic sum s(a,b)=a+b-ab	
Lukasiewicz s(a,b)=min(a+b,1)	
Drastic Sum $s(a,b)=\begin{cases} b & a=0 \\ a & b=0 \\ 1 & \text{otherwise} \end{cases}$	

The output exhibits various properties with different combinations of t-norms and t-conorms.

2.3 Fuzzy Modeling and Neurofuzzy Modeling

Fuzzy models are developed using the linguistic information granules' capability of fuzzy logic. Such models differ from other types of models because they can represent knowledge in a transparent manner via a collection of fuzzy "if-then" rules, or via an equivalent format such as fuzzy associative matrices [33]. Figure 2.2 illustrates a sample fuzzy model of a temperature controller described by a set of "if-then" rules:

if	temperature is cold	then	turn on the heater
if	temperature is normal	then	do nothing
if	temperature is hot	then	turn down the heater

Figure 2.2 Fuzzy model interpreted by extracted fuzzy rules

These "if-then" rules are usually generalized as "if antecedent then consequent."

As fuzzy models come with a clear interpretation, they are often referred to as white-box models in contrast to the previously discussed neural networks-based models which are considered to be black-box models. Essentially two main types of fuzzy model has been extensively studied, namely the Mamdani [41] model and the Takagi-Sugeno-Kang (TSK) model [82]. The difference between the Mamdani model and the TSK model lies in the consequent part of IF–THEN rules. In the TSK model, the consequents are usually calculated as linear functions of the antecedent variables, while the Mamdani model represents the consequent output by means of fuzzy sets. In essence, the Mamdani fuzzy model focuses more on interpretation and the TSK model pays more attention to the accuracy of the model.

Aside from the two main types of fuzzy model, some other popular types of fuzzy models are reported in the literature. The fuzzy decision tree [25] generalizes the decision tree to its fuzzy counterpart by admitting fuzzy sets as decision components positioned at individual nodes. The fuzzy cognitive map [81] is an extension of cognitive maps. The fuzzy expert system [15] is an expert system that applies fuzzy logic instead of Boolean logic encountered in the "traditional" expert systems.

A wide range of research and industrial areas such as electrical engineering, civil engineering, aerospace science, finance and business, medical science, etc. employ applications of fuzzy models. Examples of specific applications include analysis of electrical circuits [90], civil engineering [40, 71], modeling and analysis of financial and business performance [83], diagnosis of diseases [84], modeling software costs in software engineering [46, 47], etc.

However, building of "if-then" rules is not a trivial task. These white-box models exhibit very limited learning ability when tuning the parameters of the rules against model performance. The learning problem of fuzzy models significantly limits their application. Many techniques have been introduced to improve the learning of fuzzy models including fuzzy clustering [57], evolutionary techniques [13, 43, 54], etc. However, the accuracy of the resulting models has often been somewhat lacking.

To enable a system to avoid the drawbacks of a neural network and deal with cognitive uncertainties in a manner more like a human, one may intuitively think of incorporating the concept of fuzzy logic into a neural network. The resulting hybrid system is known as a fuzzy neural network (FNN) or neurofuzzy system [5, 7, 35, 52, 72]. The conventional neurofuzzy system underwent substantial improvement in the recent decade [39, 77]. The adaptive-network-based fuzzy inference system (ANFIS) proposed by Jang in 1993 [24] is one of the early neurofuzzy models implemented in the framework of adaptive networks. By using the hybrid learning procedure, the proposed ANFIS can construct input-output mapping in the form of fuzzy "if-then" rules. ANFIS however has strong computational restrictions. In addition, the linear representation at the consequent part of the generated rules limits the transparency of the rules to some extent.

Neurofuzzy modeling has evolved substantially in recent years; we can divide the research into two groups:

1) *Research is focused on solving the trade-off between interpretability and accuracy:*

In reviewing the referenced literature, the major difficulty of neurofuzzy modeling becomes apparent—it is the tradeoff between interpretation and accuracy. Ideally, neurofuzzy systems should fully exploit the strength of these two technologies, that is, accuracy—capability to precisely represent the real system, and interpretability—capability to express the behavior of the real system by means of rules. Unfortunately, accuracy and interpretability are always contradictory in neurofuzzy computing; thus this synergy is a target yet to be satisfied. Many hybrid approaches are analyzed to develop accurate and still-interpretable fuzzy rule-based systems including the use of weighted rules [1-3], scaling function [28], and induced expert knowledge [17].

One of the most interesting approaches is Pedrycz's fuzzy neural networks (FNN) based on fuzzy neurons [21]. Such fuzzy neurons emerge as result of a vivid synergy between fuzzy set constructs and neural networks. In essence, these neurons are functional units that retain logic aspects of processing and learning capabilities characteristic of artificial neurons and neural networks. In the setting of fuzzy neurons, the synergy of learning and transparency is well addressed. Resulting fuzzy neural networks have been discussed in detail in a number of previous studies [38, 55, 56, 58, 60, 62]. Our research is based on this type of neurofuzzy system.

2) *Research is devoted to input selection and to developing new learning approaches:*

Conventional neurofuzzy models suffer from combinatorial rule explosion [5, 26]; that is, model complexity grows exponentially as input increases. To deal with this problem, several approaches have been developed. One popular method builds a hierarchical structure [9, 12, 61, 69, 85] so that each level only deals with a small number of inputs and the total number is only linearly increased. Such a hierarchical structure is especially helpful in high dimension problems as it substantially reduces the number of rules. Many new learning approaches are also introduced in the area of neurofuzzy computing.

2.4 General Architecture of the Fuzzy Model

Although there are various types of fuzzy models presented in the literature, all these models can be summarized in the general framework illustrated in Figure 2.3. The general framework of a fuzzy model is composed of three main functional components: input interface, processing core, and output interface [55]. The input interface realizes all communication between the external physical world (say, a modeling environment) and the processing core (logic model) which operates on a higher, more abstract conceptual level. The input interface accepts any input data no matter what its format (say, continuous values or discrete values) and transforms it into the internal format of information granules as understood by the logic model. To communicate the results of logic processing back to the physical environment, a broad range of so-called defuzzification procedures [34, 53] are implemented. The architecture of the logic model itself is focused on processing information granules coming from the interface; thus the construction of interfaces is a fundamental and critical task of fuzzy modeling. The processing core is the most important component of the fuzzy model. It consists of a knowledge base that contains the structure and details of system behavior and that realizes inference through granular computation. The topology of such a core often consists of a collection of fuzzy "if-then" rules.

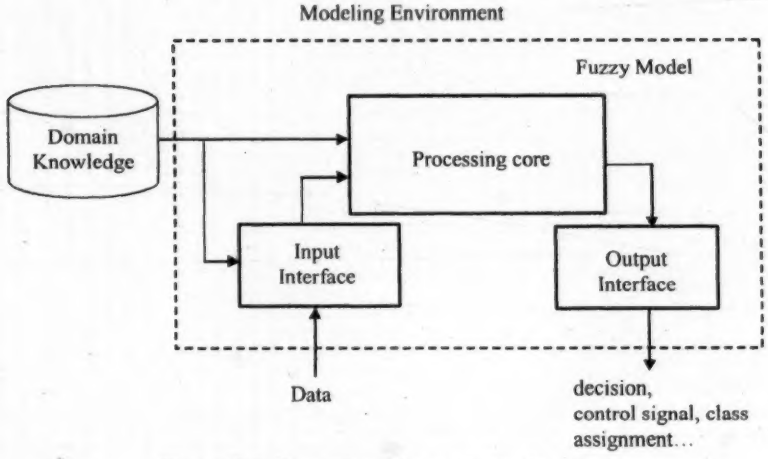


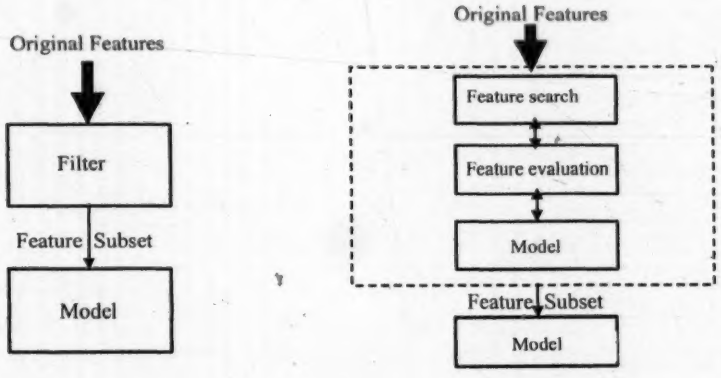
Figure 2.3 General topology of a logic model

2.5. Interfaces of a Fuzzy Model

As a fuzzy model interacts with a physical world whose manifestation does not usually arise at the level of information granulation, it becomes apparent that there is a need for some interface between the model and this world. Such interfaces are well known in fuzzy modeling. The input interface is usually composed of fuzzifiers (granular encoders) and the output interface is comprised of defuzzifiers (granular decoders) [53].

The interfaces allow interaction between the processing core of the model and the physical world outside the model. More specifically, the input interface realizes the transformation of physical world data into an internal format of information granules understood by the logic-processing core.

The input interface can be implemented by a "filter" or a "wrapper". Filter and wrapper are two fundamental strategies originating from the feature selection [6, 27, 88]. In the filter strategy, features are filtered independently of the model in a preprocessing step as shown in Figure 2.4 (a). In contrast, the wrapper strategy wraps the model and uses the model itself as part of the function evaluation of the feature subset (see Figure 2.4 (b)).



(a) Filter (b) Wrapper

Figure 2.4 Filter vs. wrapper strategies of feature subset selection

Since the filtering model is a preprocessing procedure which ignores the effect of the attribute subset on the performance of the induction algorithm, many researchers have pointed out that it may not be as effective and general as the wrapper model. However, the wrapper strategy is usually model-dependent and computationally expensive, and therefore is implemented less frequently.

In fuzzy modeling we can consider the input interface as a filter that preprocesses the data and passes the processed data into the processing core. As illustrated in Figure 2.5, there are two fundamental phases in the input interface: in the first phase each variable is granulated into a collection of semantically meaningful information granules. Discretization is a process of transforming continuous models and equations into discrete counterparts. This process is usually carried out as a first step toward making them suitable for numerical evaluation and implementation on digital computers. Hence discretization is a reasonable way to deal with information granulation. Quantization is essential for information to be processed on a digital computer. The process of quantization is the second phase in the input interface in which fuzzy granules are converted to binary format (binarization). Discretization is discussed in Chapter 5.

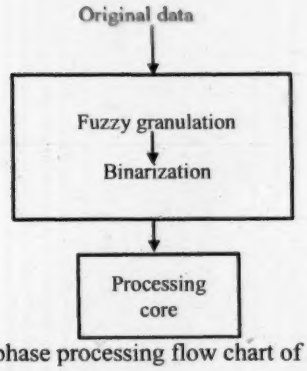


Figure 2.5 Two-phase processing flow chart of an input interface

The output interface decodes the results from the "internal" world (processing core) of the unit hypercube back to numeric values in the output space. The decoding process is discussed in Section 6.2 of Chapter 6.

2.6 Conclusions

This chapter presents a literature review of the diversity of logic-based modeling techniques including neural networks, fuzzy modeling, and neurofuzzy modeling. The aim is to provide the reader with a background to the predominant challenges in neurofuzzy modeling. Current trends in neurofuzzy modeling and limitations of current models are discussed. Among existing models, the fuzzy neuron-based model exhibits a superb learning ability and interpretability; thus it is highly applicable to the objectives outlined in this thesis. A general fuzzy modeling framework is presented and key components are discussed.

Bibliography

[1] R. Alcalá, J. Alcalá-Fdez, J. Casillas, O. Cordón, and F. Herrera, "Hybrid learning models to get the interpretability-accuracy trade-off in fuzzy modeling", Soft Computing, vol. 10, no. 9, pp. 717-734, 2006.
[2] R. Alcalá, O. Cordón, and F. Herrera, "Combining rule weight learning and rule selection to obtain simpler and more accurate linguistic fuzzy models", Modeling with Words 2003, pp. 44-63, 2003.
[3] R. Alcalá, J. Casillas, O. Cordón, and F. Herrera, "Improving simple linguistic fuzzy models by means of the weighted COR methodology", IBERAMIA 2002: 294-302.
[4] J. A. Anderson, and E. Rosenfeld, Neurocomputing: Foundations of Research, MIT Press, Cambridge, MA, 1987.
[5] R. Bellman, Adaptive Control Process, Princeton: Princeton Univ. Press, 1961.

[6] J. Bins, and B.A. Draper, "Feature selection from huge feature sets", Proceedings of the 8th IEEE International Conference on Computer Vision, vol. 2, pp. 159-165, 2001.
[7] D. Butnariu, "L-fuzzy automata: description of a neural model", Modern Trends in Cybernetics and Systems, vol. 2, pp.119–125, 1977.
[8] R. Cignoli, I. D'Ottaviano, and D. Mundici, Algebraic Foundations of Many-valued Reasoning, Dordrecht: Kluwer, 2000.
[9] F.L. Chung, and J.C. Duan, "On multistage fuzzy neural network modeling", IEEE Transactions. on Fuzzy Systems, vol. 8, pp.125-142, 2000.
[10] A-M. Cretu, E.M. Petriu, and G.G. Patry, "A comparison of neural networks architectures for geometric modeling of 3D objects", IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 155-160, 2004.
[11] A.-M. Cretu, E.M. Petriu, and G.G. Patry, "Neural network-based adaptive sampling of 3D object surface elastic properties," IEEE Instrumentation and Measurement Technology Conference, pp. 285-290, 2004
[12] J.C. Duan, and F.L. Chung, "Multilevel fuzzy relational systems: structure and identification", Soft Computing, vol. 6, pp.73-86, 2002.
[13] D. Dumitrescu, B. Lazzerini, and L. Jain, Evolutionary Computation, Boca Raton, FL, CRC Press, 2000.
[14] J. Fodor, "Left-continuous t-norms in fuzzy logic: an overview", Acta Polytechnica Hungarica, vol. 1, no. 2, 2004
[15] H. Furuta, M. Umano, K. Kawakami, H. Ohtani, and N. Shiraishi, "A fuzzy expert system for durability assessment of bridge decks", Proceedings of Uncertainty Modeling and Analysis, First International Symposiumon, pp. 522-527, 1990.
[16] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, New York, NY, Addison Wesley, 1989.
[17] S. Guillaume, and L. Magdalena, "Expert guided integration of induced knowledge into a fuzzy knowledge base", Soft Computing, vol. 10, no. 9, pp. 773-784, 2006.
[18] P. Hájek, Metamathematics of Fuzzy Logic, Dordrecht: Kluwer, 1998
[19] M.H. Hassoun, Associative Neural Memories: Theory and Implementation, New York, Oxford University Press, 1993
[20] D.O. Hebb, The Organization Of Behavior, Wiley, N.Y., 1949.
[21] K. Hirota and W. Pedrycz, "OR/AND neuron in modeling fuzzy set connectives", IEEE Transactions on Fuzzy Systems, vol. 2, pp.151-161, 1994.
[22] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", Proceedings of the National Academy of Sciences, vol. 81, pp. 3099-3092, 1984.
[23] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences, vol. 79, pp. 2554-2558, 1982.
[24] Jyh-Shing Roger Jang, "ANFIS: Adaptive-Network-based Fuzzy Inference

System", IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, pp. 665-685, 1993.
[25] C.Z. Janikow, "Fuzzy decision trees: issues and methods", IEEE Transactions on Systems Man, Cybernetics--Part B: Cybernetics, vol. 28, no, 1, pp. 1-14, 1998.
[26] Y. Jin, "Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement", IEEE Transactions on Fuzzy Systems, vol. 8, no. 2, pp. 212-221, 2000.
[27] K. Kira, and I.A. Rendell, "The feature selection problem: Traditional methods and a new algorithm", The 10th National Conference on Artificial Intelligence, MIT Press, pp. 129-134, 1992
[28] F. Klawonn, "Reducing the number of parameters of a fuzzy system using scaling functions", Soft Computing, vol. 10, no. 9, pp. 749-756, 2006
[29] E. Klement, R. Mesiar, and E. Pap, Triangular Norms, Dordrecht: Kluwer, 2000
[30] K-Y. Ko, G.G. Patry, A-M. Cretu, and E.M. Petriu, "Neural network model for wastewater treatment plant control," IEEE International Workshop on Soft Computing Techniques for Instrumentation, Measurement and Related Application, pp. 38-43, 2003.
[31] T. Kohonen, "Self-organized formation of topologically correct feature map", Biological Cybernetics, vol. 43, pp. 56-69, 1982.
[32] B. Kosko, "Bidirectional Associative Memories," IEEE Transactions on Systems, Man, and Cybernetics, vol. 18, no. 1, pp. 49-60, 1988.
[33] B. Kosko, Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence, Prentice Hall, Englewood Cliffs, N.J., 1992.
[34] B. Kosko, Fuzzy Thinking, 1993
[35] S.C. Lee, and E.T. Lee, "Fuzzy neural networks", Mathematical Biosciences, vol. 23, pp. 151-177, 1975
[36] M.T. Leung, W.E. Engeler, and P. Frank, "Fingerprint processing using backpropagation neural networks," Proceedings of the International Joint Conference on Neural Networks I, pp. 15-20, 1990.
[37] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, "Handwritten digit recognition with a back-propagation network," Advances in Neural Information Processing Systems, vol. 2, pp. 248-257, 1990.
[38] X. Liang, and W. Pedrycz, "Fuzzy logic-based networks: A study in logic data interpretation", International Journal of Intelligence Systems, vol. 21, no. 12, pp. 1249-1267, 2006.
[39] C. Mahabir, F.E. Hicks, and A.Robinson Fayek, "Neuro-fuzzy river ice breakup forecasting system", Journal of Cold Regions Science and Technology, vol. 46, no. 2, pp. 100-112, 2006.
[40] C. Mahabir, F.E. Hicks, and A. Robinson Fayek, "Application of fuzzy logic to forecast seasonal runoff", Journal of Hydrologic Processes, vol. 17, pp. 3749-3762, 2003.

[41] E.H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis", IEEE Transactions on Computers, vol. 26, pp. 1182–1191, 1977.
[42] W. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133, 1943.
[43] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, New York, NY, Springer-Verlag, 1997.
[44] M. Minsky, and S. Papert, Perceptrons, Cambridge, MA: MIT Press, 1969
[45] M. Mitchell, An Introduction to Genetic Algorithms, Cambridge, MA, MIT Press, 1998
[46] P. Musilek, W. Pedrycz, G. Succi, and M. Reformat, "Software cost estimation with granular models", Applied Computing Review, vol. 8, no. 2, pp. 24-29, 2000.
[47] P. Musilek, W. Pedrycz, G. Succi, M. Reformat, and N. Sun, "Interactive simulation of a fuzzy model for software cost estimation", Proceedings of the 2001 International Conference on Simulation and Multimedia in Engineering Education, 2001
[48] P. Musílek, "Artificial neural networks in robot navigation", Neural Network World, vol. 5, no. 5, pp. 929-944, 1995.
[49] P. Musílek, "Principles of autonomous mobile robot control", Neural Network World, vol. 3, no. 3 , pp. 249-260, 1993.
[50] P. Musilek, E. Pelikan, T. Brabec, and M. Simunek, "Recurrent neural network based gating for natural gas load prediction system", World Congress on Computational Intelligence, WCCI 2006, pp. 7127-7132, 2006.
[51] D. Parker, "Learning logic," Technical Report TR-87, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, 1985.
[52] W. Pedrycz, "Fuzzy neural networks and neurocomputations", Fuzzy Sets and Systems, vol. 56, pp. 1-28, 1993.
[53] W. Pedrycz, Fuzzy Sets Engineering, CRC Press, Boca Raton, FL, 1995.
[54] W. Pedrycz, Computational Intelligence: An Introduction, CRC Press, BocaRaton, FL, 1997.
[55] W. Pedrycz, "Heterogeneous fuzzy logic networks: Fundamentals and development studies", IEEE Transactions on Neural Networks, vol. 15, pp. 1466-1481, 2004.
[56] W. Pedrycz, "Logic-driven fuzzy modeling with fuzzy multiplexers", Engineering Applications of Artificial Intelligence, vol. 17, pp. 383-391, 2004
[57] W. Pedrycz, Knowledge-Based Clustering: From Data to Information Granules, Wiley, New York, 2005.
[58] W. Pedrycz, and G. SucciM, "fXOR fuzzy logic networks", Soft Computing, vol. 7, no. 2, pp. 115-120, 2002.
[59] W. Pedrycz, and F. Gomide, An Introduction to Fuzzy Sets: Analysis and Design, MIT Press, Cambridge, MA, 1998.

[60] W. Pedrycz, and M. Reformat, "Genetically optimized logic models", Fuzzy Sets and Systems, vol. 150, pp. 351–371, 2005.
[61] W. Pedrycz, M. Reformat, and C. W. Han, "Cascade architectures of fuzzy neural networks", Fuzzy Optimization and Decision Making, vol. 3, no. 1, pp. 5-37, 2004.
[62] W. Pedrycz, M. Reformat, and K. Li, "OR/AND neurons and the development of interpretable logic models", IEEE Transactions on Neural Networks, vol. 17, no. 3. pp. 636-658, 2006.
[63] E.M. Petriu, "Neural networks for measurement and instrumentation in virtual environments", Neural Networks for Instrumentation, Measurement and Related Industrial Applications, NATO Science Series, Series III: Computer and System Sciences, vol. 185, pp.273-290, 2003.
[64] E.M. Petriu, A.-M. Cretu, and P. Payeur, "Neural network modeling Techniques for the Real-Time Rendering of the Geometry and Elasticity of 3D Objects", The 2nd IEEE International Workshop Soft Computing Applications, pp. 11-16, 2007.
[65] E.M. Petriu, P. Payeur, and A.-M. Cretu, "Haptic human interfaces for robotic telemanipulation", Proceedings of the 4th International Symposium on Applied Computational Intelligence and Informatics, pp. 53-58, 2007
[66] E.M. Petriu, T.E. Whalen, and V.Z. Groza, "Haptic perception system for robotic tele-manipulation", INES 2002, The 6th International Conference on Intelligent Engineering Systems 2002, pp. 51-55, 2002.
[67] E.M. Petriu, S.K.S. Yeung, S.R. Das, A.-M. Cretu, and H.J.W. Spoelder, "Robotic tactile recognition of pseudorandom encoded objects", IEEE Transactions on Instrumentation and Measurement, vol. 53, no. 5, pp. 1425 - 1432, 2004.
[68] E.M. Petriu, S.K.S. Yeung, S.R. Das, and H.J.W. Spoelder, "Robotic tactile recognition of pseudo-random encoded objects," Proceedings of IMTC/2003, IEEE IMTC Instrumentation and Measurement Technology Conference, pp. 1397-1401, 2003
[69] G. V. S. Raju, J. Zhou, and R. A. Kisner, "Hierarchical fuzzy control", International Journal of Control., vol. 54, no. 5, pp. 1201–1216, 1991.
[70] I. Ratner, H.O. Ali, and E.M. Petriu, "Neural network simulation of a dielectric ring resonator antenna", Journal of Systems Architecture, vol. 44, no. 8, pp. 569-581, 1998.
[71] A. Robinson Fayek, and Z. Sun, "A fuzzy expert system for design performance prediction and evaluation", Canadian Journal of Civil Engineering, CSCE, vol. 28, no. 1, pp. 1-25, 2001.
[72] A.F. Rocha, "Neural fuzzy point processes", Fuzzy Sets and Systems, vol. 5, no.2, pp.127–140, 1981.
[73] A.F. Rocha, E. Francozo, M.I. Handler, and M.A. Balduino, "Neural languages", Fuzzy Sets and Systems, vol. 3, no. 1, pp. 11–35, 1980.
[74] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review, vol. 65, pp.

386-408, 1958.
[75] F. Rosenblatt, Principles of Neurodynamics, New York: Spartan Books, 1962.
[76] D. E. Rumelhart, and J. L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, MA, 1986.
[77] P. Rusu, E.M. Petriu, T.E. Whalen, A. Cornell, and H.J.W. Spoelder, "Behavior-based neuro-fuzzy controller for mobile robot navigation", IEEE Instrumentation and Measurement Technology Conference, pp. 1617-1622, 2002.
[78] O.G. Selfridge,"Pandemonium: a paradigm for learning", Symposium of Mechanisation of Thought Processes, National Physical Laboratory, vol. 10, pp. 513-526, 1958.
[79] R.J. Schalkoff, Artificial Neural Networks, New York, McGraw-Hill, 1997
[80] T.J. Sejnowski, B.P. Yuhas, M.H. Goldstein, and R.E. Jenkins, "Combining visual and acoustic speech signals with a neural network improves intelligibility", Advances in Neural Information Processing Systems, vol. 2, pp. 232-239, 1990.
[81] W. Stach, L. Kurgan, W. Pedrycz, and M.Reformat, "Evolutionary development of fuzzy cognitive maps", The 14th IEEE International Conference on Fuzzy Systems, vol. 25, no. 25, pp. 619 – 624, 2005.
[82] H. Takagi, and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions", Fuzzy Information, Knowledge Representation, pp. 55–60, 1983.
[83] O. Thomas, O. Adam, K. Leyking, and P. Loos, "A fuzzy paradigm approach for business process intelligence", The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, pp. 27, 2006
[84] H. Wang, M. Zhang, D. Xu, and D. Zhang, "A framework of fuzzy diagnosis", IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 12, pp. 1571-1582, 2004
[85] L.X. Wang, "Analysis and design of hierarchical fuzzy systems", IEEE Transactions on Fuzzy Systems, vol. 7, no. 5, pp. 617-624, 1999.
[86] P. J. Werbos, "Beyond regression: new tools for prediction and analysis in the behavioral Sciences", Ph.D. Dissertation, Harvard University, Cambridge, MA, 1974.
[87] B. Widrow, and M. Hoff, "Adaptive Switching Circuits", IRE WESCON Convention Record, vol. 4, pp. 96-104, 1960.
[88] D. Zongker, and A. Jain, "Algorithms for feature selection: an evaluation, Pattern Recognition", Proceedings of the 13th International Conference on Pattern Recognition, vol. 2, pp.18-22, 1996.
[89] D. W. Patterson, Artificial Neural Networks:Theory and Applications, Singapore, Prentice Hall, 1996
[90] T. Yamakawa, and T. Miki, "The Current Mode Fuzzy Logic Integrated

Circuits Fabricated by the Standard CMOS", IEEE Transactions on Computers, vol. 35, no. 2, pp. 161-167, 1986
[91] L. A. Zadeh, Fuzzy Sets, Information and Control, vol. 8, pp. 338-353, 1965.

Chapter 3
Logic-Based Neurons

In this chapter we deal with a category of neurons known as fuzzy neurons which perform the logical aggregation of inputs. Within this category there are two main types of logic-driven processing units, namely AND neurons and OR neurons, cf. [4,6-8,10-12]. We will discuss recent developments in the generalization of logic processing units, i.e., unineurons [9,11]. Given their generality, unineurons show logic processing in a new light and provide opportunities for new interpretations and versatile learning capabilities. Unineurons will also be discussed with regard to selected combinations of logic operations.

3.1. AND and OR Logic Neurons

AND and OR neurons are examples of fuzzy neurons that realize some *and*- and *or*-like logic aggregation of inputs. Their realization hinges upon the use of t-norms and t-conorms (s-norms) [3,5].

Given a collection of n inputs x_i i=1,2,...,n, denoted in a vector form $\mathbf{x} \in [0,1]^n$, the AND neuron, as shown in Figure 3.1, is a nonlinear logic processing element with the n-input **x** governed by the following relationship

$$y=\text{AND}(\mathbf{x};\mathbf{w}) \tag{3.1}$$

where **w** represents an n-dimensional vector of adjustable connections (or weights) whose values are in the unit interval. The composition of **x** and **w** is realized by a certain t-s composition that makes use of some t- and t-conorms.

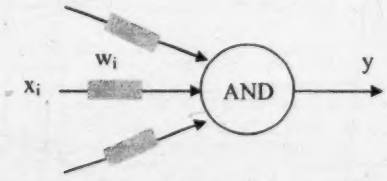


Figure 3.1 Logic processing of neurons: AND neuron

The aggregation consists of two steps. In the first step inputs are combined with the corresponding weight. In the second step we aggregate these partial results

with the aid of some t-norm. In other words we obtain

$$y=AND(\mathbf{x};\mathbf{w})=\mathop{T}_{i=1}^{n}[w_i s x_i] \tag{3.2}$$

Here "s" and "t" stand for the t-conorms and the t-norms, respectively. We can rewrite (3.1) in an equivalent format that facilitates its further interpretation

$$y = (x_1)_{w_1} \text{ and } (x_2)_{w_2} \cdots \text{and} (x_n)_{w_n} \tag{3.3}$$

where $(x_i)_{w_i}$ stands for a shorthand notation for (x_i or w_i), $i = 1,....n.$

The AND neuron realizes logic operations on the input values, and the connections **w** play an important role: the connections are to adjust the impact that individual input may have on the aggregation results. In particular, when setting the AND neuron with the connection **w** = 0, expression (3.3) can be further reduced to the "pure" type of *and*-like aggregation,

$$y = x_1 \text{ and } x_2 \text{ and } .... \text{ and } x_n$$

Similarly when connection **w** = 1 we get y =1.

Thus for the AND neuron, in general, the lower the value of the connection, the more essential the corresponding input. In limit, we view the input meaningless if the associated connection is equal to 1.

By reverting the order of the t- and t-conorms in (3.2), we arrive at the construct of an OR neuron

$$y=OR(\mathbf{x};\mathbf{w})=\mathop{S}_{i=1}^{n}[w_i t x_i] \tag{3.4}$$

Figure 3.2 illustrates such aggregation offered by the OR neurons.

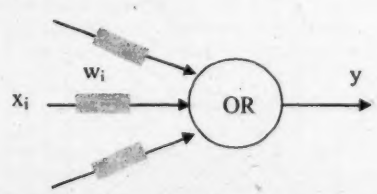


Figure 3.2 Logic processing of neurons: OR neuron

For interpretation purposes, we adhere to the notation

$$y = [x_1]_{w_1} \text{ or } [x_2]_{w_2} \cdots \text{or } [x_n]_{w_n} \tag{3.5}$$

where $[x_i]_{w_i}$ denotes (x_i and $\hat{w}_i$) i=1,....n.

Similarly, when the connection of the OR neuron **w** = 1, the expression (3.5) is reduced to

$$y = x_1 \quad \text{or} \quad x_2 \quad \text{or} \quad \ldots \quad \text{or} \quad x_n$$

when the connection **w** = 0, $y = 0$.

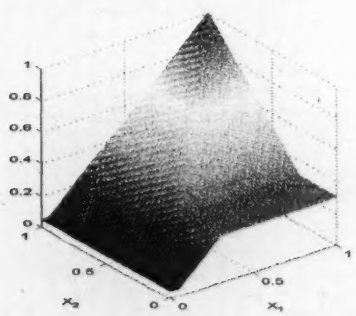
The above expressions indicate that the higher the value of the connection, the more relevant the corresponding input. In limit, if $w_i = 0$, the i-th input x_i can be fully eliminated.

3.2 Characteristics of AND and OR Logic Neurons

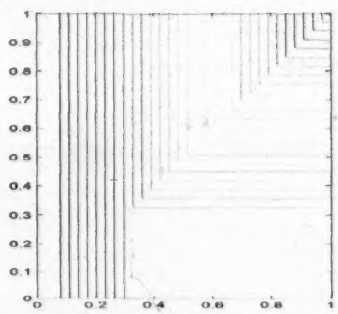
As discussed in the previous section, AND and OR neurons exhibit nonlinear characteristics which are inherently implied by the use of t- and s-norms. To visualize such characteristics, we consider a two-dimensional feature space (x_1-x_2) so that each neuron has only two input variables.

3.2.1 Input-output characteristics of AND neurons

Figure 3.3 (a)-(f) and Figure 3.4 (a)-(f) show a diversity of input-output characteristics of the AND neuron. Note that the characteristics are affected by the use of triangular norms and the values of the connections.



(a)



(b)

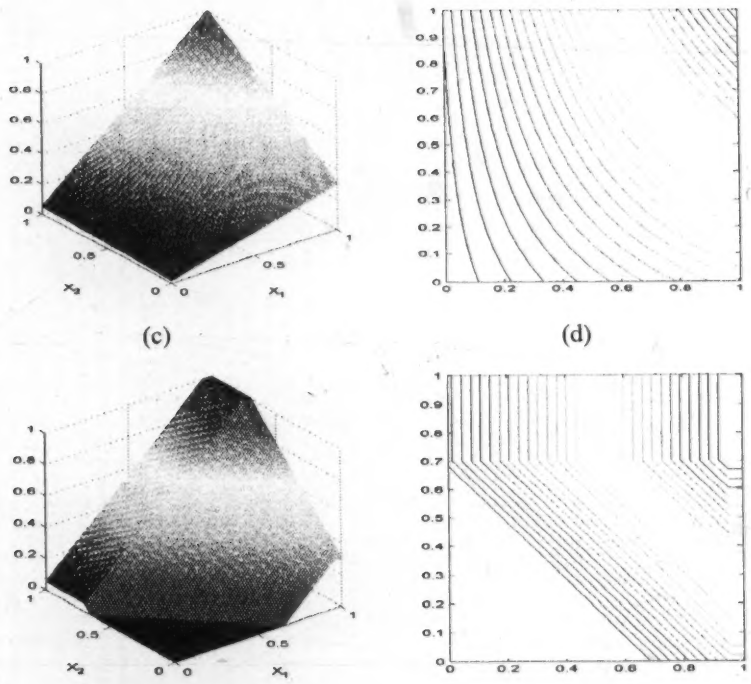
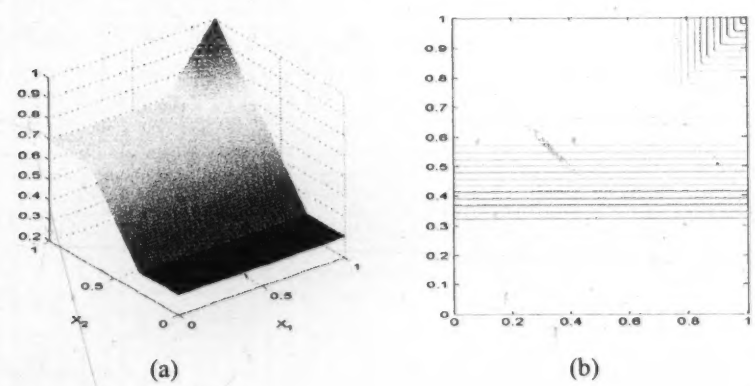


Figure 3.3 Input–output characteristics of an AND neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.05 0.30]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives.



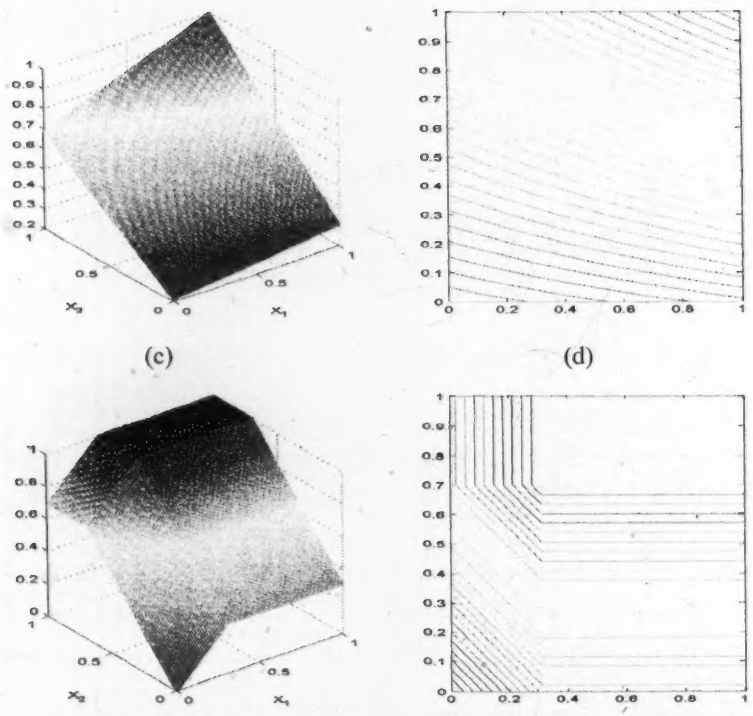


Figure 3.4 Input–output characteristics of an AND neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.69 0.30]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives.

3.2.2 Input-output characteristics of OR neurons

Figure 3.5 (a)-(f) and Figure 3.6 (a)-(f) show a diversity of input-output characteristics of the OR neuron. Again note that the characteristics are affected by the use of triangular norms and by the values of the connections.

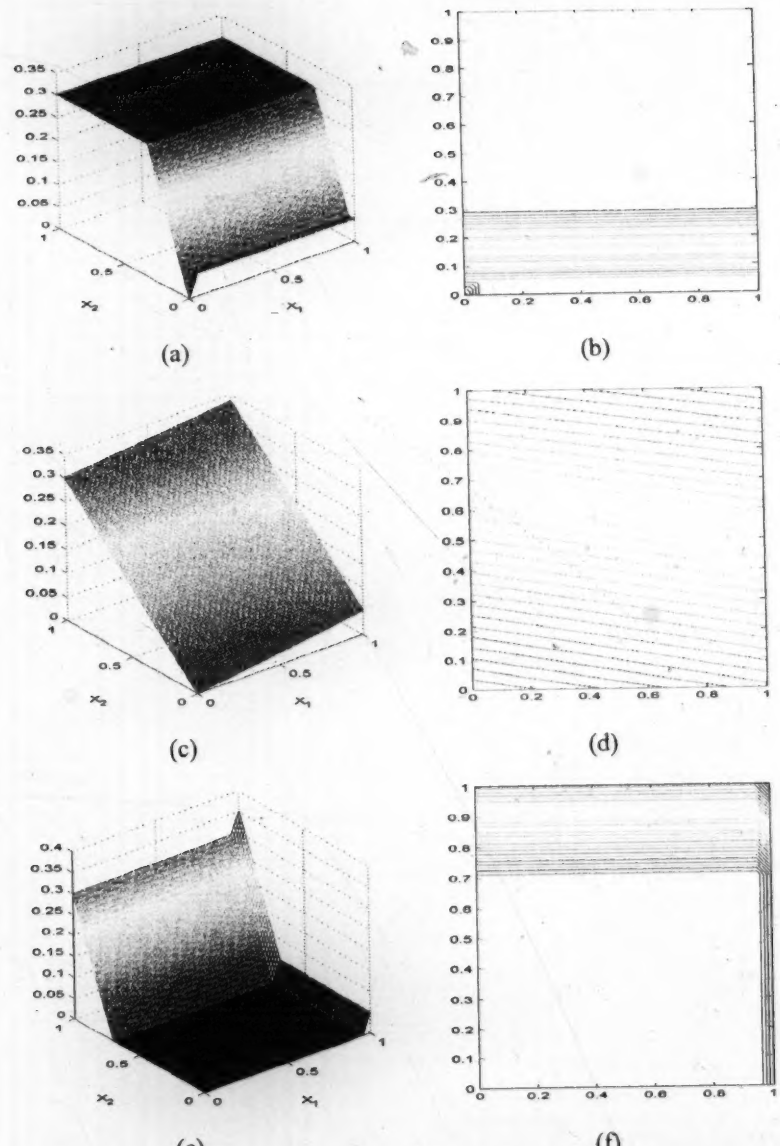


Figure 3.5 Input–output characteristics of an OR neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.05 0.30]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives.

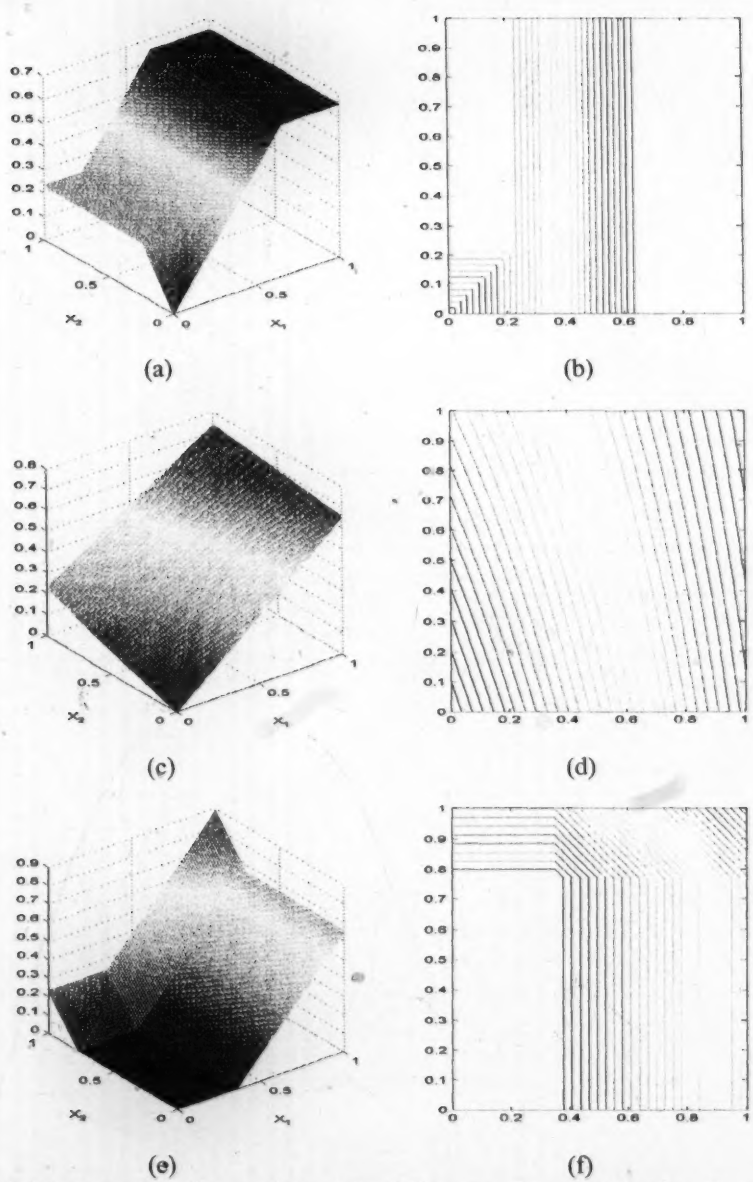


Figure 3.6 Input–output characteristics of an OR neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to **w** = [0.65 0.23]. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives

From the above figures, two things become clear. First, the connections help us control the form of the input–output characteristics and they deliver all the

necessary plasticity; this shows their important role in any learning activity. Second, various t-norms and t-conorms are used to adjust the input-output mapping to conform to available experimental data.

3.3 Fuzzy AND and OR unineurons

Before proceeding with a description of fuzzy unineurons, we will briefly revisit the fuzzy uninorm.

Uninorms form hybrids of t- and t-conorms by binding the two standard logic operators encountered in logic and fuzzy sets [1-2, 13-15]. One such uninorm is a mapping u: $[0, 1]^2 \rightarrow [0, 1]$ which satisfies properties of commutativity, monotonicity, and associativity.

Commutativity	$u(x, y) = u(y, x)$
Monotonicity	$u(x, y) \geq u(z, v)$ for $x > z$ and $y > v$
Associativity	$u(x, u(y, z)) = u(u(x, y), z)$

More importantly, by introducing the identity element "g" which varies between 0 and 1, we can implement switching between the "and" and "or" properties of the logic operators. For instance, given input x and identity element g, $u(x, g) = x$, $g \in [0,1]$. So when $g = 0$ we end up with the "or" type of aggregation, $u(x, 0) = x$, and when $g = 1$, it returns the "and" type of aggregation, namely $u(x, 1) = x$.

In the literature, there are many types of realizations for uninorms. Here, we choose the following equation which is logic meaningful.

$$u(x,y,g) = \begin{cases} gt(\frac{x}{g}, \frac{y}{g}) & x, y \in [0, g] \\ g + (1-g)s(\frac{x-g}{1-g}, \frac{y-g}{1-g}) & x, y \in [g, 1] \\ \min(x, y) & \text{otherwise} \end{cases} \tag{3.6}$$

or

$$u(x,y,g) = \begin{cases} gt(\frac{x}{g}, \frac{y}{g}) & x, y \in [0, g] \\ g + (1-g)s(\frac{x-g}{1-g}, \frac{y-g}{1-g}) & x, y \in [g, 1] \\ \max(x, y) & \text{otherwise} \end{cases} \tag{3.7}$$

In equations (3.6) and (3.7), x and y are two inputs between 0 and 1; t and s are the t-norm and s-norm defined previously. Figure 3.7 illustrates the computation

of a uninorm as expressed in equation (6).

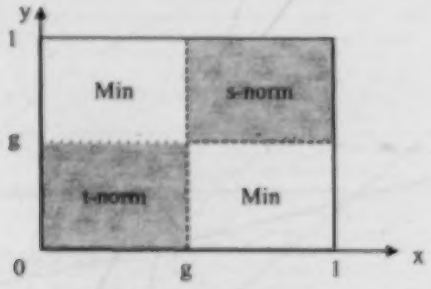


Figure 3.7 Computing of uninorms

From Figure 3.7, we note that identity point g splits the unit square into four regions, with t-norm and s-norm located diagonally. When g = 1, the uninorm becomes the t-norm: u(x, y) = t(x, y), and when g = 0, the uninorm becomes s-norm: u(x, y) = s(x, y).

As discussed in the previous subsection, incorporating uninorms into fuzzy neurons turns them into unineurons[9,11]. Like general neurons, unineurons are treated as n-input nonlinear static processing units that map elements in the unit hypercube $[0, 1]^n$ into elements in the unit interval of [0, 1]. There are two levels of logic processing carried out in the processing units. More specifically, given a collection of inputs $x = [x_1\ x_2 \ldots x_n]$ and the parameters of unineurons, including connections $w = [w_1\ w_2 \ldots w_n]$ and identity points $g = [g_1\ g_2 \ldots g_n]$, at the first level we exploit the use of uninorms by combining individual input x_i with corresponding connections w_i and identity points g_i giving rise to the expression $u(x_i; w_i, g_i)$; the resulting aggregation is realized at the second level.

Two fundamental categories of logic neurons are introduced here which will be referred to as AND unineurons and OR unineurons, or AND_U and OR_U in shorthand notation.

AND unineurons (AND_U)

As shown in Figure 3.8 (a), given a collection of "n" inputs $\mathbf{x} = [x_1\ x_2 \ldots x_n]$ and the parameters of unineurons including connections $\mathbf{w} = [w_1\ w_2 \ldots w_n]$ and identity points $\mathbf{g} = [g_1\ g_2 \ldots g_n]$, the AND_U neurons process them in the form of

$$y = \text{AND_U}(\mathbf{x}; \mathbf{w}, \mathbf{g}) \qquad (3.8)$$

or in coordinates it can be rewritten as

$$y = \overset{n}{\underset{i=1}{T}}(u(x_i; w_j, g_i)) \tag{3.9}$$

In the above equation, the uninorm operation is realized in the form governed by the uninorm equation while T(t) denotes a certain t-norm. As becomes apparent, the name of the unineuron is implied by the "and" type of aggregation of the individual inputs. Moreover, the standard AND neurons are subsumed by the AND_U neurons when using the zero vector of the identity points g = 0, that is:

$$y = \overset{n}{\underset{i=1}{T}}(u(x_i; w_j, 0)) = \overset{n}{\underset{i=1}{T}}(s(x_i; w_j)) \tag{3.10}$$

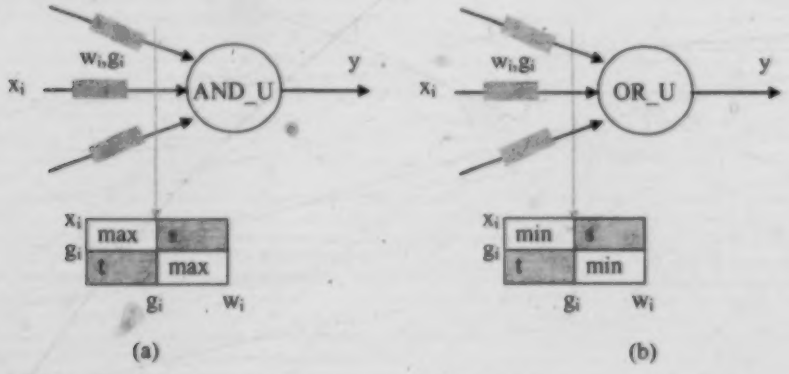


Figure 3.8 Logic processing of unineurons (a) AND unineuron and (b) OR unineuron

OR unineuron (OR_U)

Similarly, an n-input single output g realized by this processing unit

$$y = OR_U(\mathbf{x}; \mathbf{w}, \mathbf{g}) \tag{3.11}$$

represents an or–type of aggregation of the partial results produced by the uninorm combination of the corresponding inputs, see Figure 3.8 (b). We can rewrite the above equation as

$$y = \overset{n}{\underset{i=1}{S}}(u(x_i; w_j, g_i)) \tag{3.12}$$

where S(s) stands for any s-norm (t-conorm). Also, the standard OR neurons are subsumed by the OR_U neurons when entries of the identity points g are all ones (g = 1), that is,

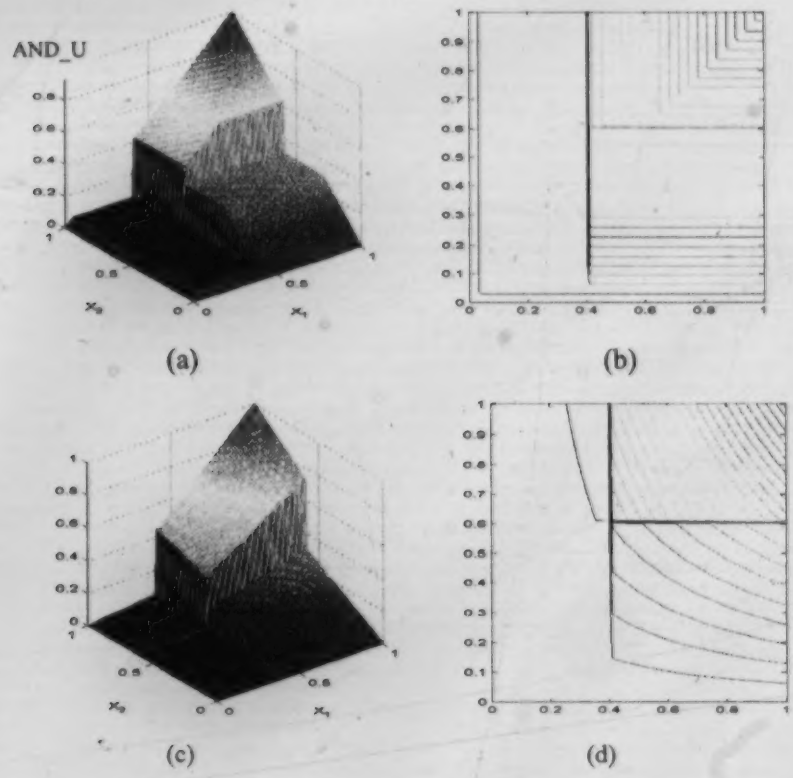
$$y = \overset{n}{\underset{i=1}{S}}(u(x_i; w_j, 1)) = \overset{n}{\underset{i=1}{S}}(t(x_i; w_j)). \tag{3.13}$$

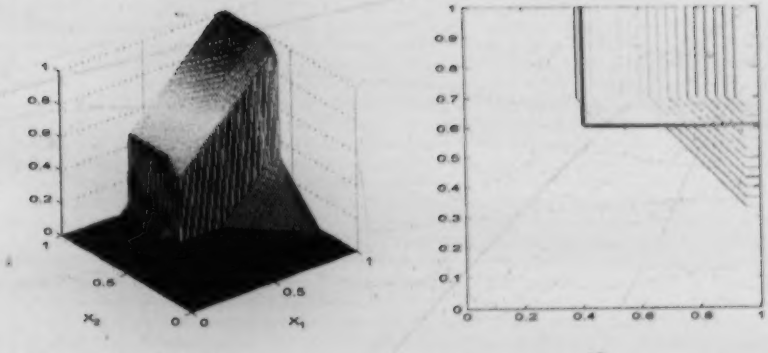
3.4 Characteristics of AND and OR unineurons

Here we examine the nonlinear characteristics of AND and OR unineurons. To simplify for illustration, we still consider each neuron as having only two input variables.

3.4.1 Input-output characteristics of AND unineurons

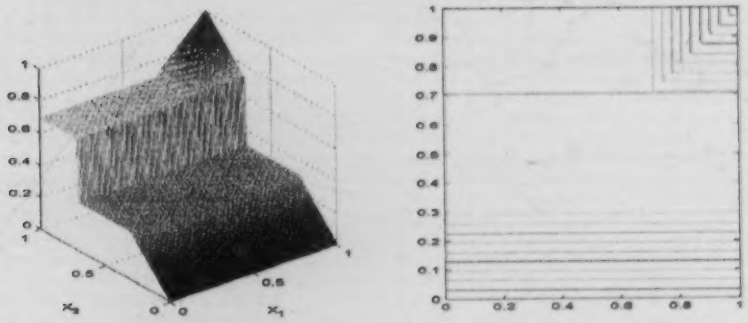
Figure 3.9 (a)-(f) and Figure 3.10 (a)-(f) show a diversity of input-output characteristics of the AND unineuron. Note that aside from the use of some norms which change the values of the connections, the characteristics of unineurons are also affected by the values of the identity points.



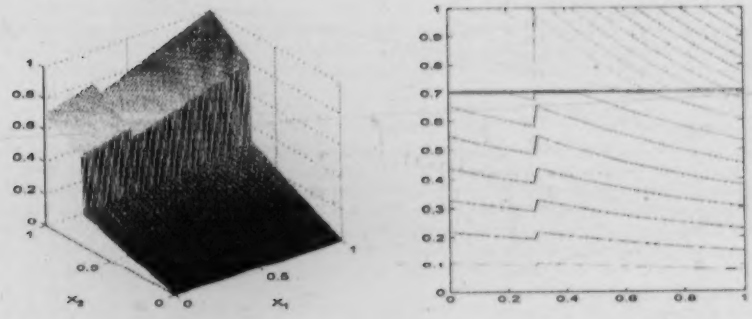


(e) (f)

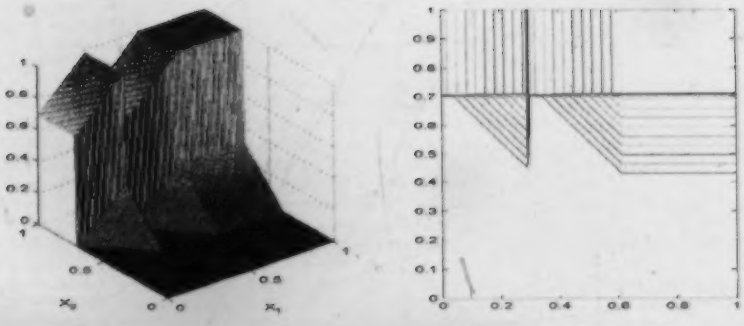
Figure 3.9 Input–output characteristics of AND unineurons for selected pairs of t- and s-norms. In all cases, the corresponding connections and identity points are set to **w** = [0.05 0.30] and g = [0.40 0.60], respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives.



(a) (b)



(c) (d)

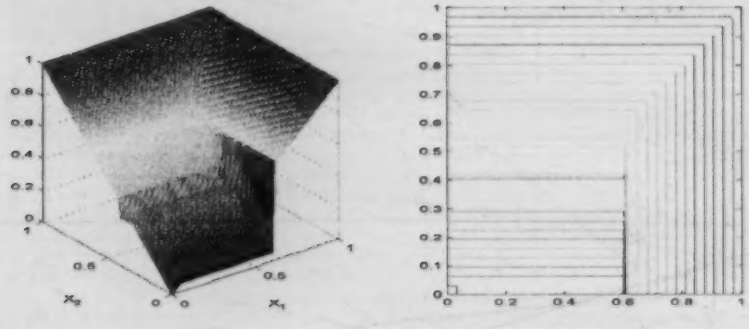


(e) (f)

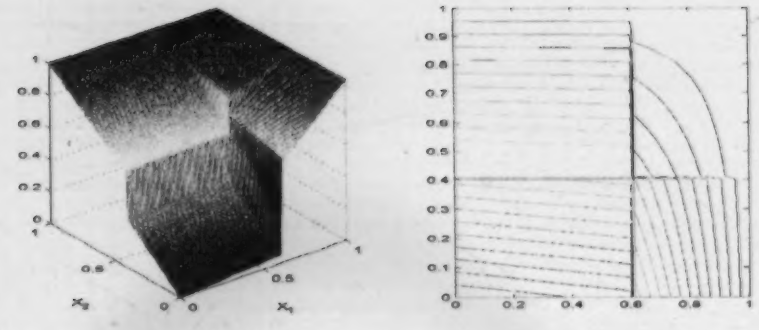
Figure 3.10 Input–output characteristics of AND unineurons for selected pairs of t- and s-norms. In all cases, the corresponding connections and identity points are set to **w** = [0.69 0.30] and g = [0.30 0.70], respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives.

3.4.2 Input-output characteristics of OR unineurons

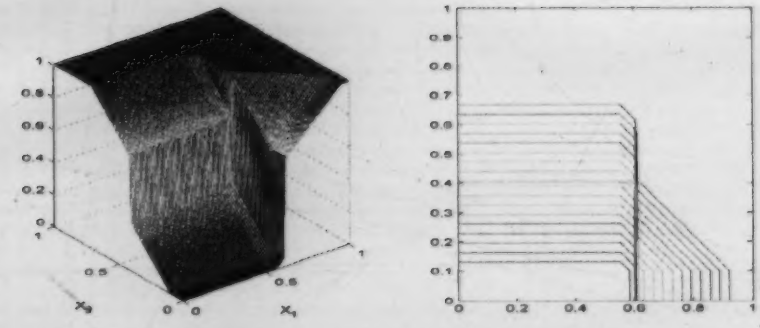
Figure 3.11 (a)-(f) and 3.12 (a)-(f) show a diversity of input-output characteristics of the OR unineuron.



(a) (b)

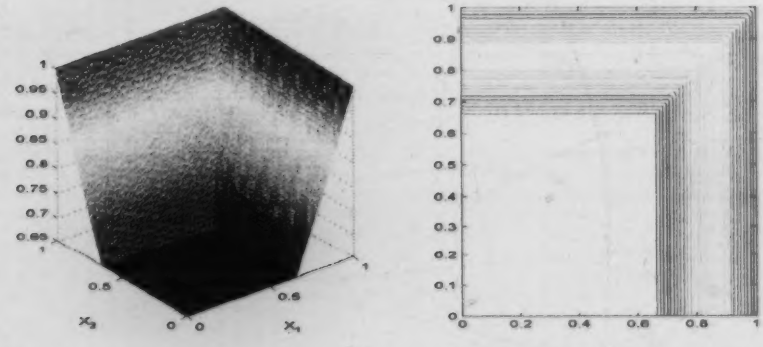


(c) (d)



(e) (f)

Figure 3.11 Input–output characteristics of OR unineurons for selected pairs of t- and s-norms. In all cases, the corresponding connections and identity points are set to **w** = [0.05 0.30] and **g** = [0.40 0.60], respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives.



(a) (b)

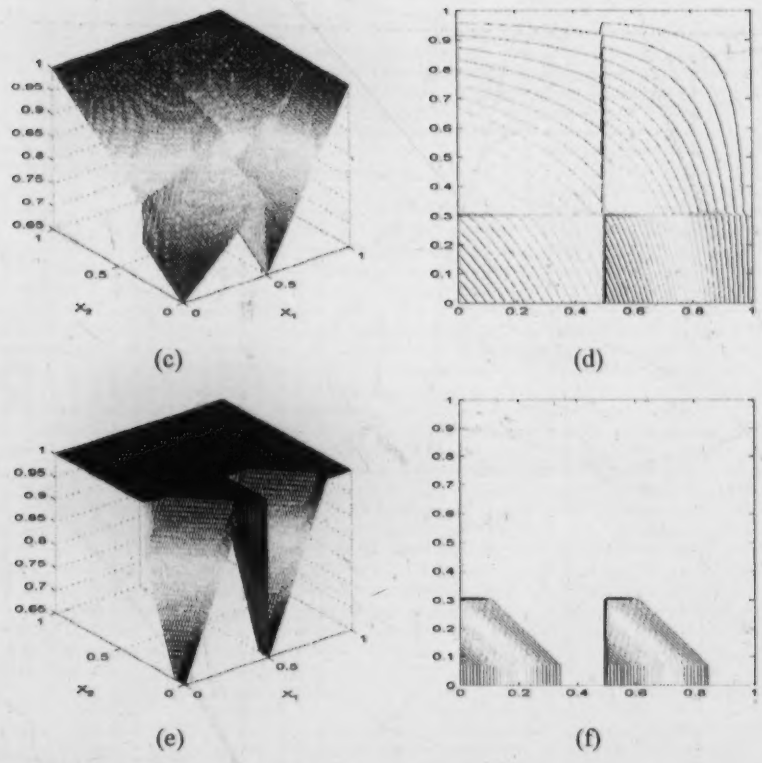


Figure 3.12 Input–output characteristics of an AND unineuron for selected pairs of t- and s-norms. In all cases, the corresponding connections and identity points are set to $\mathbf{w} = [0.65\ 0.23]$ and $\mathbf{g} = [0.50\ 0.30]$, respectively. Both 3D plots (left) and contour plots (right) are used here to visualize the characteristics. (a, b) min and max; (c, d) product and probabilistic sum; (e, f) Lukasiewicz *and* and *or* connectives.

With the setting of the identity point in an AND unineuron to zero, i.e., $\mathbf{g} = \mathbf{0}$, the AND unineuron becomes an AND neuron. In particular, with the setting of $\mathbf{g} = \mathbf{0}$ in the example shown in Figure 3.9, the characteristics will be represented exactly the same as the plots shown in Figure 3.3. Similarly, the same processing effect occurs when dealing with the OR unineurons. By setting the g of an OR unineuron to one, i.e., $\mathbf{g} = \mathbf{1}$, the OR unineuron is turned into an OR neuron. Figure 3.5 shows the input-output characteristics of an OR unineuron with the identity point $\mathbf{g} = \mathbf{1}$.

The nonlinear character of input–output dependencies can be seen very clearly. Such nonlinear character depends upon the specific t-norms and conorms involved. The connections and identity points impact the neurons in a direct manner. Thus, the advantage of unineurons resides with the significant flexibility

and learning capabilities offered by the connections, identity points, and triangular norms. Even though the neuron itself may look a bit complicated, its underlying logic expression becomes straightforward and readable.

3.5 Conclusions

The neurons discussed in this chapter accept the logic type of aggregation of inputs, and are thus categorized as aggregative neurons. Fuzzy neurons emerge as a result of a vivid synergy between fuzzy set constructs and neural networks. In essence, given that logic operators are used in the development of logic neurons, logic neurons retain logic aspects of processing and learning capabilities characteristic of artificial neurons and neural networks. Such interesting functional properties can be beneficial when designing networks formed with logic neurons.

Bibliography

[1] T. Calvo, and R. Mesiar, "Continuous generated associative aggregation operators", Fuzzy Sets Systems, vol. 126, pp. 191–197, 2002.

[2] J.C. Fodor, R.R. Yager, A. Rybalov, "Structure of uninorms", International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 5, pp. 411–427, 1997.

[3] P. Hájek, Metamathematics of Fuzzy Logic, Kluwer, Dordrecht, 1998

[4] K. Hirota, W. Pedrycz, "OR/AND neuron in modeling fuzzy set connectives", IEEE Transactions on Fuzzy Systems, vol. 2, pp. 151–161, 1994.

[5] E.P. Klement, R. Mesiar, and E. Pap, Triangular Norms, Kluwer, Dordrecht, 2000.

[6] W. Pedrycz, Fuzzy Sets Engineering, CRC Press, Boca Raton, FL, 1995.

[7] W. Pedrycz, "Heterogeneous fuzzy logic networks: fundamentals and development studies", IEEE Transactions on Neural Networks, vol. 15, pp. 1466–1481, 2004.

[8] W. Pedrycz, "Fuzzy neural networks and neurocomputations", Fuzzy Sets and Systems, vol. 56, pp. 1–28, 1993.

[9] W. Pedrycz, "Logic-based fuzzy neurocomputing with unineurons", IEEE Transactions on Fuzzy Systems, vol.14, no. 6, pp. 860-873, 2006.

[10] W. Pedrycz, and F. Gomide, An Introduction to Fuzzy Sets: Analysis and Design. MIT Press, Cambridge, MA, 1998.

[11] W. Pedrycz, and K. Hirota, "Uninorm-based logic neurons as adaptive and interpretable processing constructs", Soft Computing, vol. 11, no.1, pp. 41-52, 2007.

[12] W. Pedrycz, M. Reformat, and K. Li, "OR/AND neurons and the

development of interpretable logic models", IEEE Transactions on Neural Networks, vol. 17, no 3. pp. 636-658, 2006.
[13] R.R. Yager, and A. Rybalov, "Uninorm aggregation operators", Fuzzy Sets and Systems, vol. 80, pp.111–122, 1996
[14] R.R. Yager, "Uninorms in fuzzy systems modeling", Fuzzy Sets and Systems, vol. 122, pp. 167–175, 2001
[15] R.R. Yager, "Defending against strategic manipulation in uninorm-based multi-agent decision making", European Journal of Operational Research, vol. 141, pp. 217–232, 2002

Chapter 4
Architecture of Logic-Based Networks

In this chapter, we introduce and discuss several highly effective fuzzy modeling frameworks based on the logic-based neurons covered in the previous chapter. During the design of these frameworks, we are guided by two main objectives. First, we would like to achieve a substantial level of flexibility so that the models can be easily and effectively adjusted to experimental data by a suitable learning algorithm, such as e.g., gradient-based optimization. Second, we wish to assure a high level of interpretability of the model, such that it could be easily understood by the user. With this regard, we will take advantage of the logic nature and well-defined semantics of fuzzy neurons which once learned can be easily translated into a collection of well-structured and transparent logic expressions. Finally, we will describe the details of the underlying design issues and present the key technologies involved here.

4.1 The topology of AND and OR neuron-based neural networks

In Boolean algebra, Boolean function can be expressed in a canonical form as "sum of minterms" (SOM) or "product of maxterms" (POM) [2,7,8,9]. In essence, this is usually an aggregation of several simple compound logic expressions which provides a sound insight into the simplification of these logic functions. Logic-based neurons, AND and OR neurons, aggregate the inputs and realize some quantified logic expression. Furthermore by arranging AND and OR neurons in layers, we can arrive at the generic logic realization of the well known representation scheme being used in the realization of Boolean functions.

In the case of the SOM representation, from the perspective of the architectural realization of such logic expressions we end up with two layer logic networks as shown in Figure 4.1.

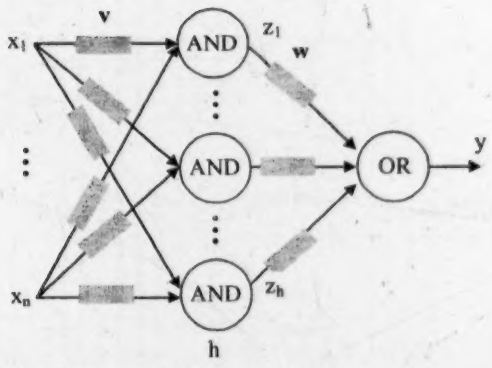


Figure 4.1 A topology of the SOM type logic network

The network is uniquely characterized by such parameters as the number of inputs (n), number of AND nodes located in the hidden layer (h). The connections of the AND neurons are organized in some matrix **V** and the vector of the connections of the OR neuron is denoted here by **w**. The network can be written in the form

$$\begin{cases} z_j = \text{AND}(\mathbf{x}, \mathbf{v}_j) & j = 1,2,...,h \\ y = \text{OR}(\mathbf{z}, \mathbf{w}) \end{cases} \tag{4.1}$$

where **x** is the input vector ($\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$), **z** is a vector of outputs of the AND neurons ($\mathbf{z} = [z_1, z_2, \cdots, z_h]^T$) and $\mathbf{v}_j$ denotes the j-th column of the connection matrix **V**.

Our intent is to visualize a diversity of the input-output characteristics of the network that arises when changing the values of the connections and combination of logic operators. For illustrative purposes we consider a two-input (x_1, x_2) SOM network with two AND neurons at the hidden layer so that each AND neuron has only two input variables. In this particular case the overall network can be described as follows

$$\text{First layer:} \quad \begin{cases} z_1 = \text{AND}(\mathbf{x}, \mathbf{v}_1) \\ z_2 = \text{AND}(\mathbf{x}, \mathbf{v}_2) \end{cases} \tag{4.2}$$

$$\text{Output layer:} \quad y = \text{OR}(\mathbf{z}, \mathbf{w}) \tag{4.3}$$

where $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2]$ is the connection matrix, **x** is the input vector $\mathbf{x} = [x_1\ x_2]^T$, and $\mathbf{z} = [z_1\ z_2]^T$ is the output vector of the AND neurons at the first layer. y is the overall aggregation of the logic behaviors and is completed by the OR neuron located at the output layer.

The plots of the input–output characteristics of a two-input-single-output structure are illustrated in Figure 4.2, Figure 4.3 and Figure 4.4 (a)-(f). Here we consider three collections of triangular norms and conorms, namely min-max, product probabilistic sum, and Lukasiewicz t-norm and t-conorm (s-norm).

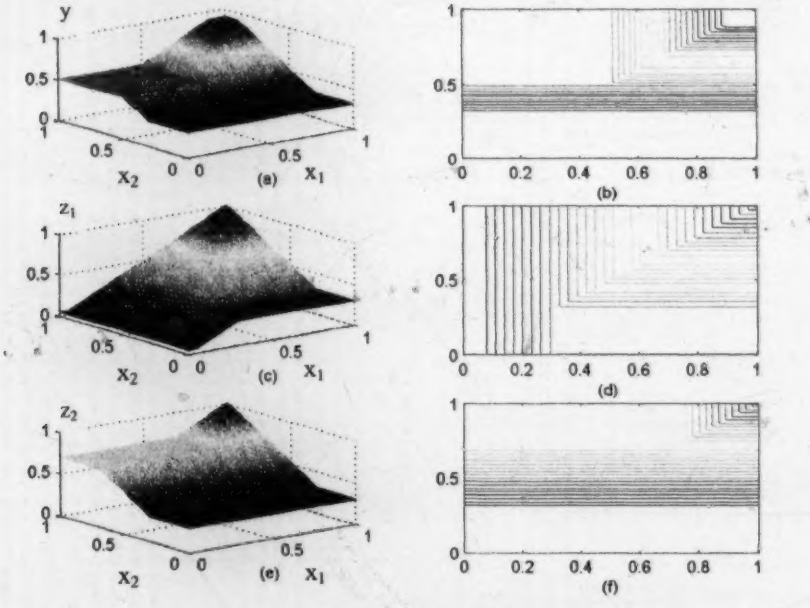


Figure 4.2 Characteristics of SOM network for selected values of the connections and realizations of t- and t-conorms(min and max)
First AND neuron in the hidden layer **w** = [0.05 0.30]
Second AND neuron in the hidden layer **w** = [0.69 0.30] ,
OR unineuron at output layer **w** = [0.90 0.50]
(a) 3D plot of network input-output (x_1-x_2-y) (b) 2D contour plot of network output y
(c) 3D plot of the first AND neuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second AND neuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

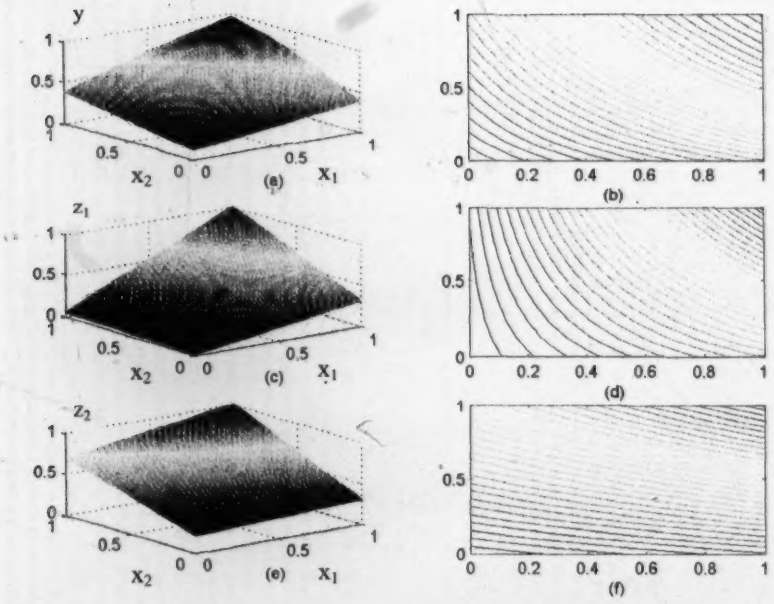


Figure 4.3 Characteristics of SOM network for selected values of the connections and realizations of t- and t-conorms(product and probabilistic sum)
First AND neuron in the hidden layer **w** = [0.05 0.30]
Second AND neuron in the hidden layer **w** = [0.69 0.30]
OR unineuron at output layer **w** = [0.90 0.50]
(a) 3D plot of network input-output (x_1-x_2-y) (b) 2D contour plot of network output y
(c) 3D plot of the first AND neuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second AND neuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

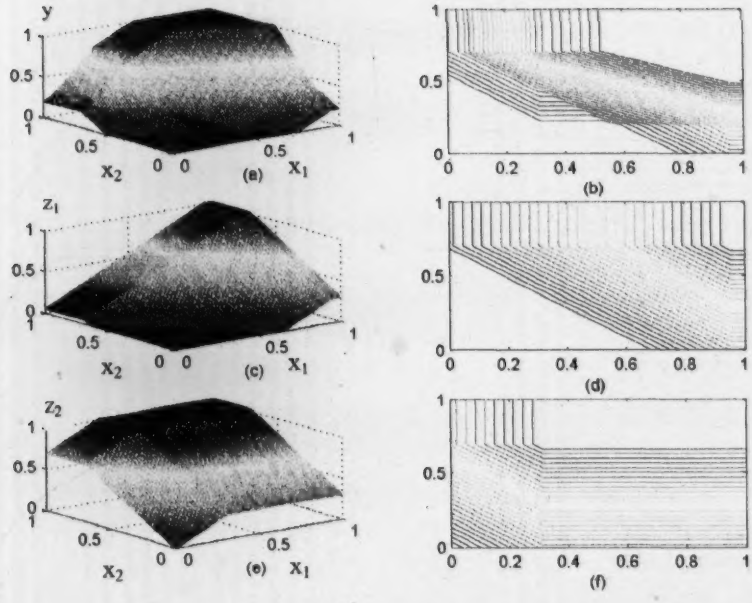


Figure 4.4 Characteristics of SOM network for selected values of the connections and realizations of t- and t-conorms (Lukasiewicz t and s-norm)
First AND neuron in the hidden layer **w** = [0.05 0.30]
Second AND neuron in the hidden layer **w** = [0.69 0.30]
OR unineuron at output layer **w** = [0.90 0.50]
(a) 3D plot of network input-output (x_1-x_2-y) (b) 2D contour plot of network output y
(c) 3D plot of the first AND neuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second AND neuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

By inspecting the figures, we conclude that the network exhibits highly nonlinear processing ability depending upon the specific realizations of the logic connectives. By using different combinations of t- and s-norms we realize a substantial level of plasticity whose usage becomes critical when learning the networks involving these neurons.

In particular, by switching the AND and OR neurons in the SOM network, we can end up with the other equivalent topology that represents the POM version of Boolean function; refer to Figure 4.5. In essence, the behavior of the OR neurons in the hidden layer is aggregated (weighted) by using the outputs of AND neurons located in the output layer.

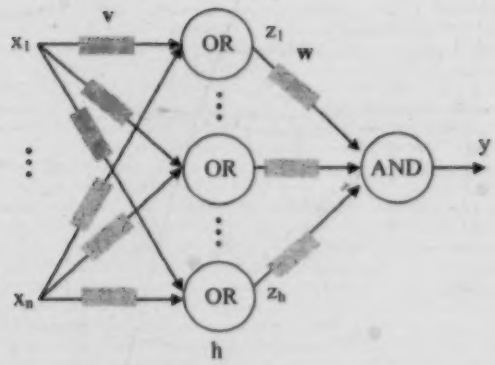


Figure 4.5 A topology of the POM type logic network

Given this functional structure, the detailed formulas of the overall network are as follows:

$$\begin{cases} z_j = \mathrm{OR}(\mathbf{x}, \mathbf{v}_j) & j = 1,2,\ldots,h \\ y = \mathrm{AND}(\mathbf{z}, \mathbf{w}) \end{cases} \tag{4.4}$$

where $\mathbf{x}$ is the input vector ($\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$), $\mathbf{z}$ is an vector of outputs of the OR neurons ($\mathbf{z} = [z_1, z_2, \cdots, z_h]^T$) and $\mathbf{v}_j$ denotes the j-th column of the connection matrix $\mathbf{V}$.

Figure 4.6 – 4.8 illustrate the characteristics of POM networks with selected t- and s-norms and specific numeric values of the connections.

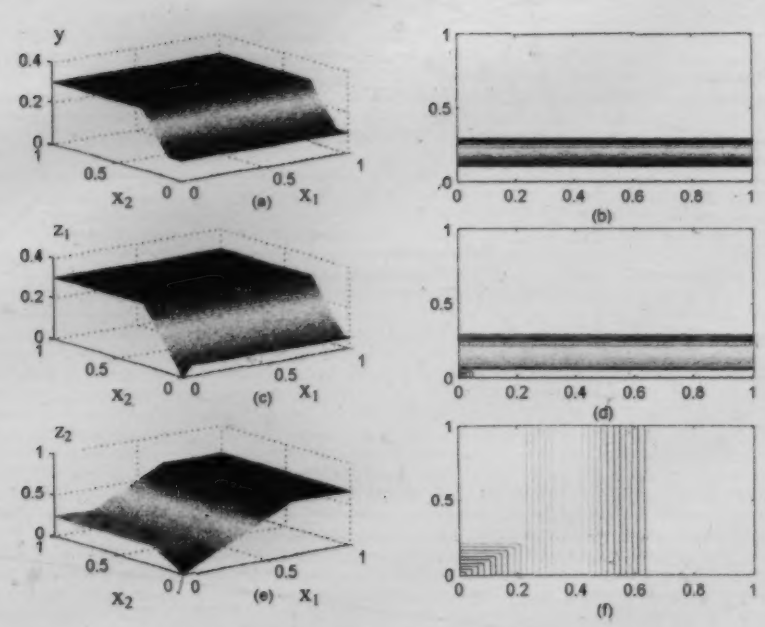


Figure 4.6 Characteristics of POM network for selected values of the connections and realizations of t- and t-conorms (min and max)
First OR neuron in the hidden layer **w** = [0.05 0.30].
Second OR neuron in the hidden layer **w** = [0.65 0.23]
AND unineuron at output layer **w** = [0.10 0.60]
(a) 3D plot of network input-output (x_1-x_2-y) (b) 2D contour plot of network output y
(c) 3D plot of the first OR neuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second OR neuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

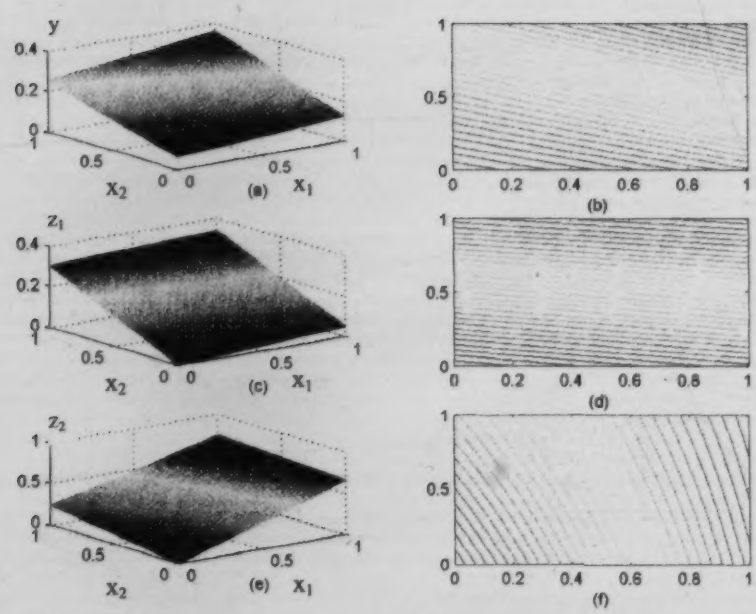


Figure 4.7 Characteristics of POM network for selected values of the connections and realizations of t- and t-conorms (product and probabilistic sum)
First OR neuron in the hidden layer **w** = [0.05 0.30]
Second OR neuron in the hidden layer **w** = [0.65 0.23]
AND unineuron at output layer **w** = [0.10 0.60]
(a) 3D plot of network input-output (x_1-x_2-y) (b) 2D contour plot of network output y
(c) 3D plot of the first OR neuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second OR neuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

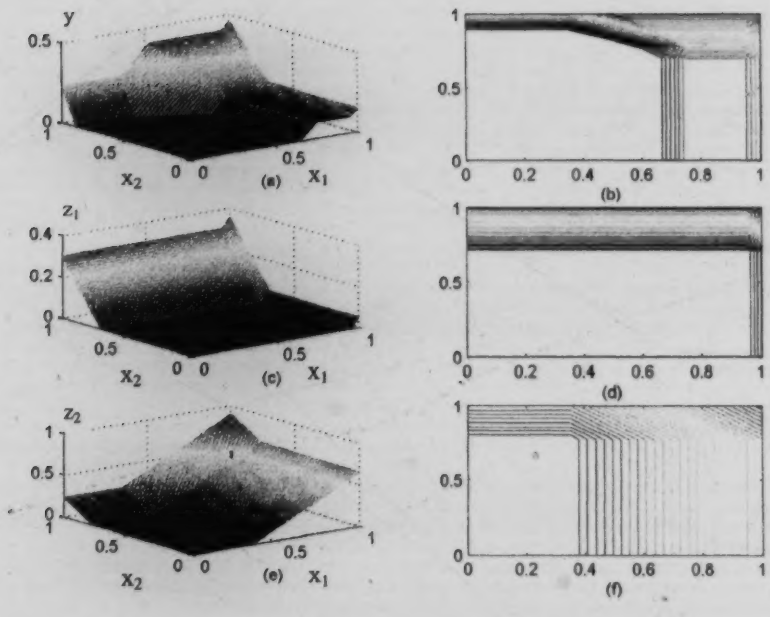


Figure 4.8 Characteristics of POM network for selected values of the connections and realizations of t- and t-conorms (Lukasiewicz t and s-norm)
First OR neuron in the hidden layer **w** = [0.05 0.30]
Second OR neuron in the hidden layer **w** = [0.65 0.23]
AND unineuron at output layer **w** = [0.10 0.60]
(a) 3D plot of network input-output (x_1-x_2-y) (b) 2D contour plot of network output y
(c) 3D plot of the first OR neuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second OR neuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

4.2 The topology of AND and OR unineuron-based neural networks

As for the architecture of unineuron-based logic networks, we consider the same topology by considering a two-layer structure in which the first layer consists of a collection of AND unineurons whereas the processing at the second layer is realized by means of OR unineurons, see Figure 4.9.

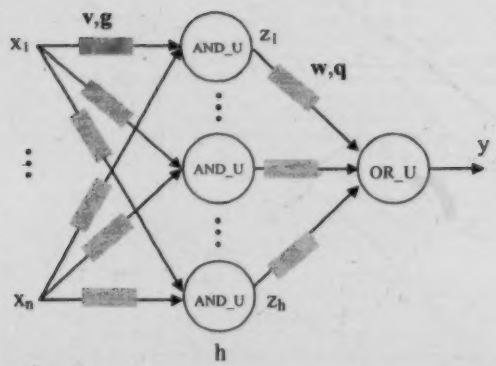


Figure 4.9 A topology of a multiple-input-single-output SOM network formed by means of AND_U and OR_U neurons

This figure illustrates the parameters of the logic network that become available in the design of the network. In addition to the adjustable number (h) of AND_U neurons (processing units in the hidden layer), the connections and neutral points of the neurons are organized in matrices of connections **v** and **w** and **g** and **q**, respectively. The network can be written as

$$\begin{cases} z_j = \text{AND_U}(\mathbf{x}, \mathbf{v}_j, \mathbf{g}_j) \quad j=1,2,\ldots,h \\ y = \text{OR_U}(\mathbf{z}, \mathbf{w}, \mathbf{q}) \end{cases} \tag{4.5}$$

Figure 4.10 - 4.12 illustrate the input-output characteristics of the network for several selected combinations of values of connections and triangular norms of the neurons.

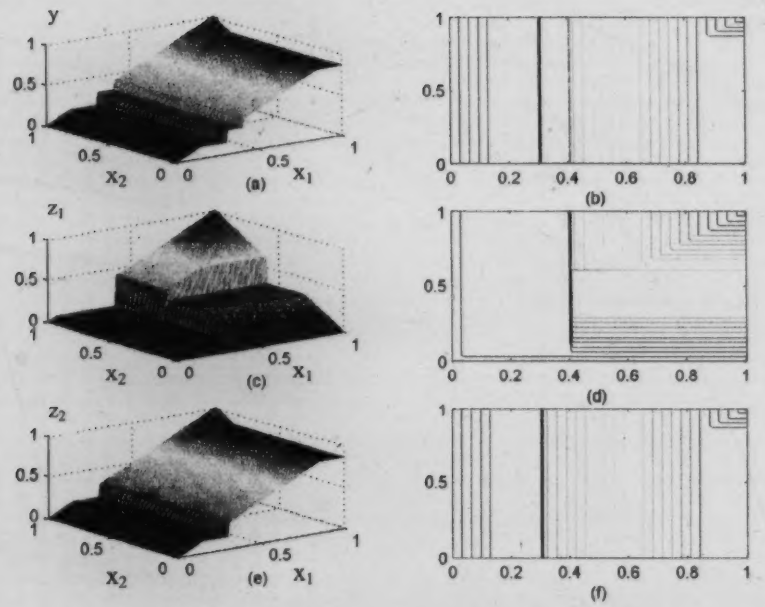


Figure 4.10 Characteristics of unineuron-based network for selected values of the connections and realizations of t- and t-conorms(min and max)
First AND unineuron in the hidden layer **w** = [0.05 0.30] **g** = [0.40 0.60]
Second AND unineuron in the hidden layer **w** = [0.15 0.85] **g** = [0.30 0.70]
OR unineuron at output layer **w** = [0.05 0.30] **g** = [0.60 0.40]
(a) 3D plot of network input-output (x_1-x_2-Y) (b) 2D contour plot of network output Y
(c) 3D plot of the first AND unineuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second AND unineuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

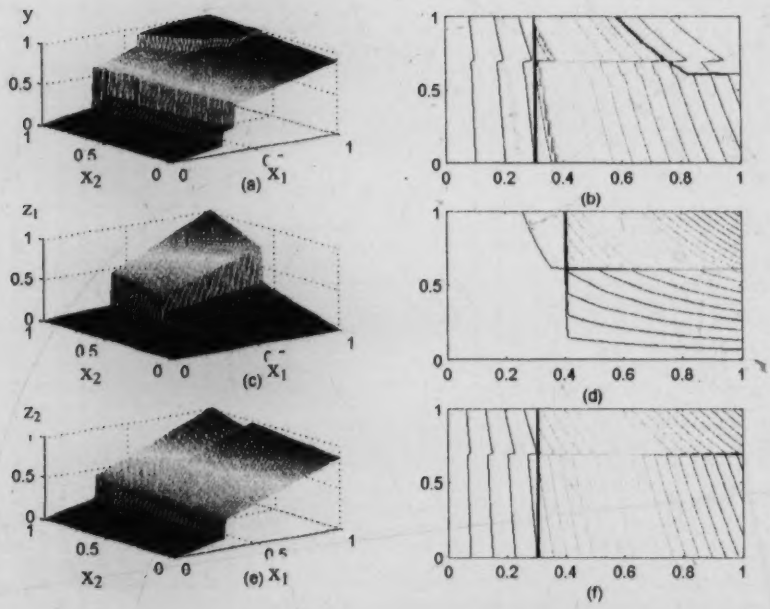


Figure 4.11 Characteristics of unineuron-based network for selected values of the connections and realizations of t- and t-conorms(product and probabilistic sum)
First AND unineuron in the hidden layer **w** = [0.05 0.30] **g** = [0.40 0.60]
Second AND unineuron in the hidden layer **w** = [0.15 0.85] **g** = [0.30 0.70]
OR unineuron at output layer **w** = [0.05 0.30] **g** = [0.60 0.40]
(a) 3D plot of network input-output (x_1-x_2-Y) (b) 2D contour plot of network output Y
(c) 3D plot of the first AND unineuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second AND unineuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

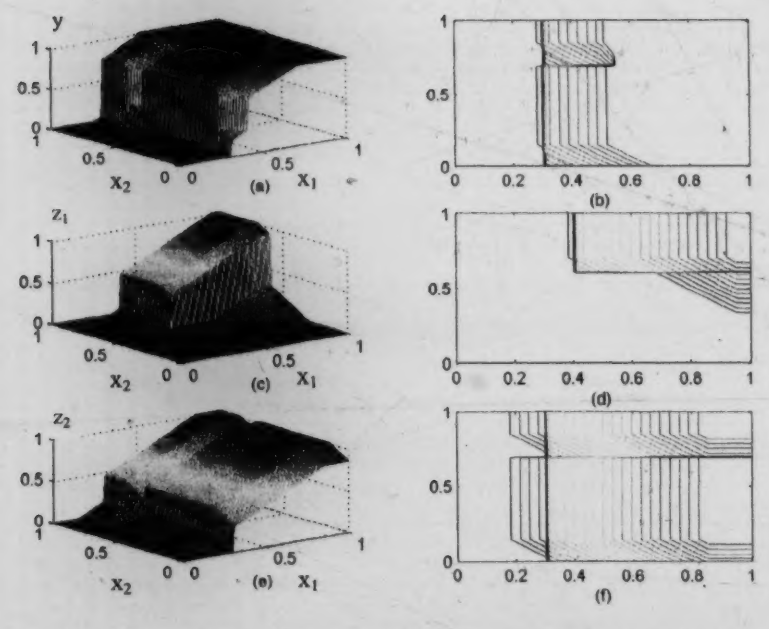


Figure 4.12 Characteristics of unineuron-based network for selected values of the connections and realizations of t- and t-conorms (Lukasiewicz t and s-norm)
First AND unineuron in the hidden layer **w** = [0.05 0.30] **g** = [0.40 0.60]
Second AND unineuron in the hidden layer **w** = [0.15 0.85] **g** = [0.30 0.70]
OR unineuron at output layer **w** = [0.05 0.30] **g** = [0.60 0.40]
(a) 3D plot of network input-output (x_1-x_2-Y) (b) 2D contour plot of network output Y
(c) 3D plot of the first AND unineuron input-output (x_1-x_2-z_1) (d) 2D contour plot of intermediate output z_1
(e) 3D plot of the second AND unineuron input-output (x_1-x_2-z_2) (f) 2D contour plot of intermediate output z_2

When we set the values of the neutral point vectors $\mathbf{g} = \mathbf{0}$ and $\mathbf{q} = \mathbf{1}$, the networks built by AND_U and OR_U neurons converts into the networks constructed by AND and OR neurons. In other words, networks built by fuzzy neurons are subsumed by the networks built by fuzzy unineurons.

Similarly, we can have equivalent POM version of the network built by fuzzy unineurons, see Figure 4.13.

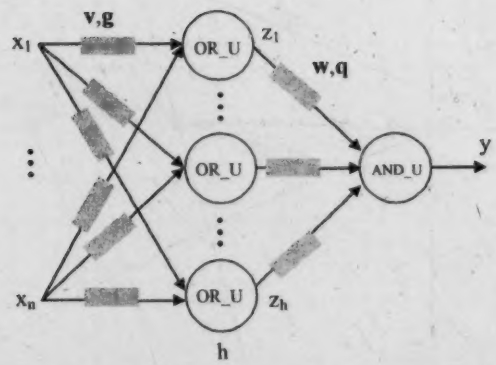


Figure 4.13 A topology of a multiple-input-single-output POM network formed by means of OR_U and AND_U neurons

4.3 Interpretation of logic networks

The logic networks come with clearly defined semantics. Evidently, the model stands for a one-to-one correspondence with its logic fabric. As illustrated in Figure 4.14, the SOM network comes with "h" AND nodes, each of which contains "n" inputs. In particular, each node in the network comes with a straightforward interpretation: as the AND node, the "n" inputs are weighted and "and"-wisely aggregated as "n" conditions. Similarly, the OR node weighs the "h" outputs from the AND nodes and aggregates them by means of "or" operations.

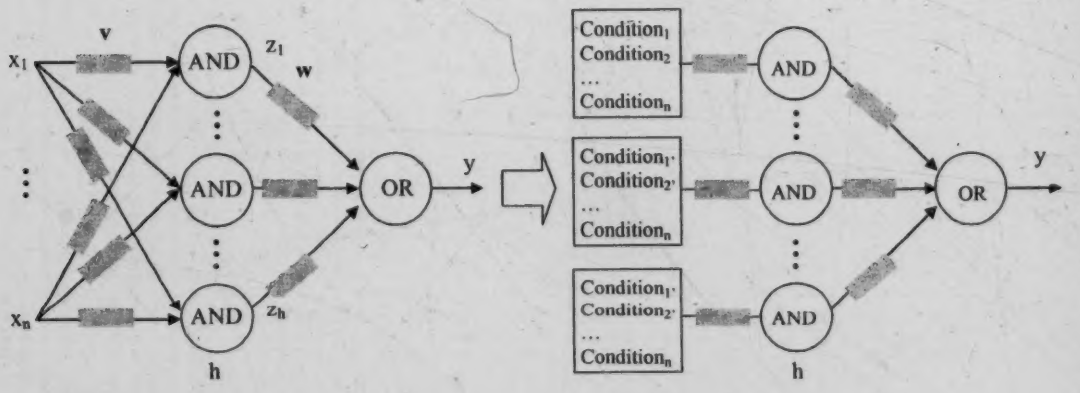


Figure 4.14 Structure of a fuzzy model represented as an aggregation of fuzzy rules

Subsequently, the network can be written as a collection of composite "if-then" rules

if

condition_1 and condition_2 and... and ... condition_n

or

$\text{condition}_1'$ and $\text{condition}_2'$ and... and ... condition_n

or

...

The learning endows the neurons with numeric connections and their values are useful in some further reduction of the network in this way improving its interpretability. Let us recall that higher values of the connections of the OR neurons are more essential while the connections with lower values could be dropped. The opposite situation occurs for AND neurons: here the higher values of the connections could be viewed as meaningless and therefore dropped. Owing to the monotonicity property, we can proceed with pruning of the weakest connections by considering the following relationships.

In particular, for AND neurons the weakest connections (which are those above some threshold) are converted to 1

$$v_\mu = \begin{cases} v & v \le \mu \\ 1 & \text{otherwise} \end{cases} \tag{4.6}$$

For the OR neuron we use the following relationship by bringing the values of the weakest connections to zero

$$w_\lambda = \begin{cases} w & w \ge \lambda \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

where $\mu, \lambda \in [0,1]$; v, w denote some connections of the neurons. By changing the values of the threshold μ and λ, we can affect the intensity of the pruning of the overall network.

The previous "h" rules then can be reduced to the new h' (h' $\le$h) "if-then" rules as follows.

if

condition_1 and condition_2 and... and ... condition_L

or

condition$_1$, and condition$_2$, and... and ... condition$_L$,

or

...

where L, L'... $\leq$ n

In these expressions, the subconditions in each rule are arranged starting with the lowest value of the connections of the AND neuron. The rules themselves are organized starting with the highest values of the OR neuron. The pruning could be completed in two different ways

(a) by applying some thresholding mechanisms. For example, by accepting some threshold values λ and μ for OR and AND neurons, respectively, we eliminate all connections whose values are below λ (OR neurons) and above μ (AND neurons)

(b) by admitting some allowable structural complexity of the logic description. Accepting a maximal number of conditions and rules, we eliminate "weaker" rules and conditions produced by the network

In general, the pruning of networks constructed by unineurons takes two steps, the first step is pruning the identity points so that some AND_U and OR_U turn into AND and OR neurons, then pruning the resulting AND and OR neurons. As the uninorm, the lower g is, the more of a t-conorm property it becomes. The higher value of g, the more essential the t-norm property it is associated. Bearing such properties in mind, we can introduce a certain threshold $\eta \in [0,1]$ that can turn the OR_U into OR neurons.

$$g_{i\eta} = \begin{cases} 1 & g_i \geq \eta \\ g_i & \text{otherwise} \end{cases} \quad i=1, \ldots, n \tag{4.8}$$

where $\mathbf{g} = [g_1\ g_2 \ldots g_n]$ is the neutral points of OR_U.

Thus, when all the entries of the identity points g of a unineuron are all ones ($\mathbf{g} = \mathbf{1}$) or can be treated as one when its value is larger than a certain threshold η, we can regard the OR_U as an OR neuron.

Similarly, we can introduce threshold $\gamma \in [0,1]$ which makes

$$g_\gamma = \begin{cases} g & g \geq \gamma \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

where $\mathbf{g} = [g_1\ g_2 \ldots g_n]$ is the vector of identity points of AND_U. When all of the entries of the neutral points g of a unineuron are zeros ($\mathbf{g} = \mathbf{0}$) or have the values lower than the threshold γ, we can turn the AND_U into a AND neuron.

After the above two-step pruning process, the network can still be rewritten and generalized as a collection of composite "if-then" rules with the ranking of connections (descent order) of the OR_U at the output layer. Although the rules itself may look a bit complicated, the logic expression is still straightforward and readable.

4.4 The design of the network

In this section, we will focus on the key design phases of the network, namely (1) forming contexts in the output space through the mechanism of fuzzy equalization; (2) context-based clustering applied to the input variables; (3) projection and reduction for each variable.

4.4.1 Formation of contexts through fuzzy equalization

The underlying idea is to construct fuzzy sets in such a way that they come with clearly defined semantics and are experimentally justifiable. Fuzzy equalization [6] helps construct linguistic labels (fuzzy sets) that are both semantically and experimentally legitimate. With the fuzzy equalization completed in the output space, we end up with "p" contexts (fuzzy sets). We assume that these fuzzy sets are described by triangular membership functions with an overlap of 0.5 between two successive linguistic terms. Furthermore, we denote the family of fuzzy sets as $\mathbf{A} = \{A_1, A_2, \ldots, A_p\}$. Assume that the probability density function (pdf) of this output is given by p(y). We can then start the lower bound of y (denoted by y_{min}) and allow the parameters of each fuzzy set to be computed as shown in Table 4.1; for more details refer to[6].

Table 4.1 Fuzzy equalization for triangular fuzzy sets A1, A2, ..., Ap

Shown above are formulas for the parameters of the corresponding fuzzy sets

$A_1(a,m,b)$	a	y_{min}
	m	$\int_{y_{min}}^{m} A_1(y)p(y)dy = \frac{1}{2p}$
	b	$\int_{m}^{b} A_1(y)p(y)dy = \frac{1}{2p}$
$A_2(a,m,b)$	a	m of A_1
	m	b of A_1

	b	$\varepsilon = \int_m^b A_1(y)p(y)dy \quad \int_m^b A_2(y)p(y)dy = \frac{1}{p} - \varepsilon$
...	...	...
$A_p(a,m,b)$	a	m of A_{p-1}
	m	b of A_{p-1}
	b	y_{max}

Note that for the discrete data set $\mathbf{Y}=\{y_1, y_2,\ldots, y_N\}$, the calculations for the probability of A, P(A) is computed through the summation of the discrete probability values.

4.4.2 Conditional Fuzzy C-Means in the formation of the blueprint of the logic network

Conditional (context-based) Fuzzy C-Means was introduced in [3,4,5] as a certain modification of the generic Fuzzy C-Means (FCM)[1] which is guided by an auxiliary (conditional) variable. This method reveals a structure within a family of data by considering their vicinity in a feature space along with the similarity of the associated values assumed by a certain conditional variable. The algorithmic underpinnings go as follows. Assume that $\mathbf{x}_1$, $\mathbf{x}_2$,..., $\mathbf{x}_N$ are n-dimensional data defined in $\mathbf{R}^n$, and we have been provided with "p" contexts (say, context-1, ..., context-p) being the result of the fuzzy equalization described in the previous section. Figure 4.15 summarizes the complete algorithm.

Given: dataset $\{\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_N\} \subset \mathbf{R}^n$
membership values for context-j: f_k (k=1, 2, ..., N, j=1, ... , p)

Defined: the number of prototypes c ($1<c<N$), exponential weight m($1<m<\infty$), the termination criterion ε ($\varepsilon>0$), and the distance function $\|\cdot\|$

Initialization Randomly initialize partition matrix U:
$U^{(0)}=[u_{ik}]$ (i=1, ..,c, k=1,...N)

Processing
Iterate iter = 1, 2, ... and compute

---

prototypes v_i: $$v_i = \frac{\sum_{k=1}^{N} u_{ik} x_k}{\sum_{k=1}^{N} u_{ik}^m}$$

partition matrix U: $$u_{ik} = \frac{f_k}{\sum_{l=1}^{c} \left(\frac{\|x_k - v_i\|}{\|x_k - v_l\|} \right)^{2/(m-1)}}$$

Until $\| U^{(iter+1)} - U^{(iter)} \| < \varepsilon$

Results prototypes and partition matrix

Figure 4.15 Conditional Fuzzy C-Means: a flow of computing

4.4.3 Projection and reduction of input variables

As becomes obvious, for each context in the output space, we have generated "c" corresponding prototypes (clusters) in the input space. Thus for "p" contexts we end up with c*p prototypes (clusters) as schematically displayed in Figure 4.16.

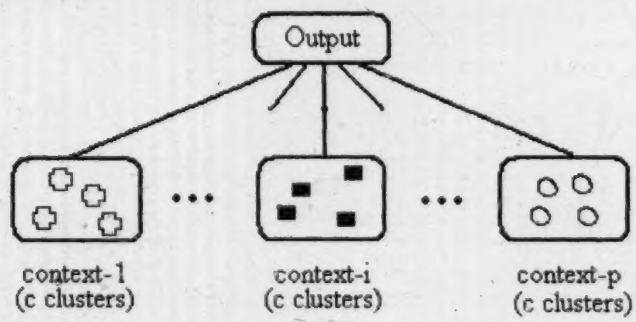


Figure 4.16 A concept of the context-based clustering; note that each context induces "c" clusters in the input space

Each prototype is then projected onto the individual variables of the input space. Along with the minimum and maximum values of each variable, the coordinates of the prototypes in the corresponding input space form c*p+2 fuzzy sets. As an example, for two contexts with two clusters per context, a relationship between the fuzzy sets of contexts and the resulting fuzzy sets arising for each input is visualized in Figure 4.17.

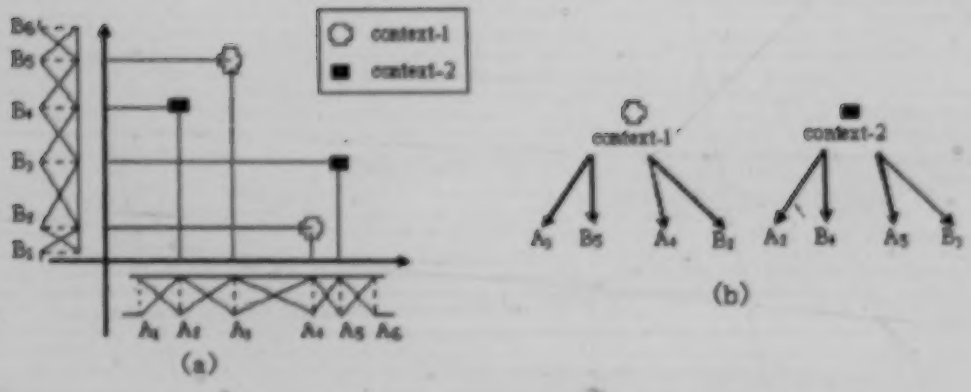


Figure 4.17 Projection of prototypes along with the contexts and induced clusters in the input space

We merge fuzzy sets if their modal values are close to each other (which make these fuzzy sets quite indistinguishable). The merging is guided by the following merging criterion.

Consider a certain input variable, say x. Its lower and upper bound are denoted as min and max, respectively. The coordinates of "c*p" prototypes result in c*p+2 fuzzy sets $\{A_1, A_2, \ldots, A_{c*p+2}\}$ built in the region of [min, max]. Define D as a threshold measure ($D = \frac{max - min}{c \cdot p}$), evaluate the distance of any two successive fuzzy sets. We merge two successive fuzzy sets by making a single fuzzy set with a trapezoidal membership function if the distance satisfies the following merging criterion:

$$\|A_{i+1} - A_i\| < \varepsilon \cdot D \qquad i = 1, 2, \ldots, c \cdot p + 1 \qquad (4.10)$$

where $\varepsilon \in [0, 1]$, and $\|\cdot\|$ is the distance function.

For instance, as a result of such merging of Figure 4.17, instead of two triangular fuzzy sets A_4 and A_5 we obtain a single trapezoidal fuzzy set C_1. Given this, we update the relationships replacing the fuzzy sets that have been merged by their new generalized version, see Figure 4.18 for more details. The structures shown in Figure 4.18(b) are then used as a blueprint to form the fuzzy logic network. Each prototype inside the context represents an AND neuron, aggregated by an OR neuron to form the output as the corresponding context.

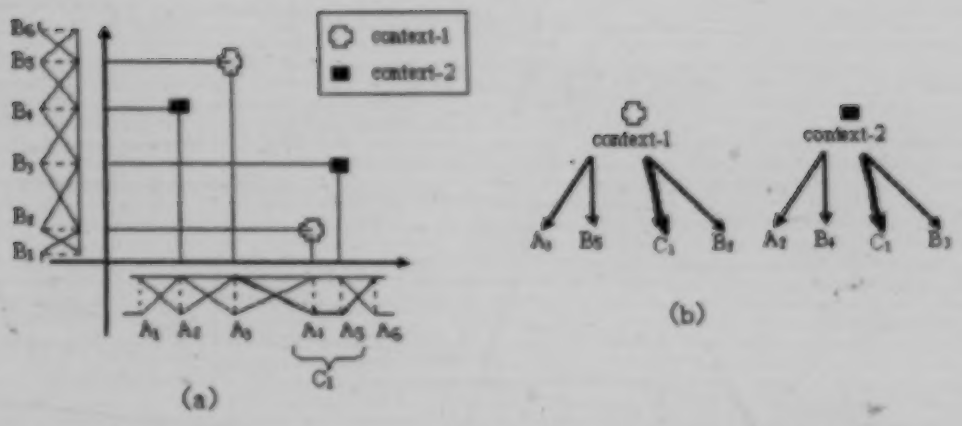


Figure 4.18 Merging of close fuzzy sets and construct of contexts using updated fuzzy sets

Note that, with such projecting and reduction measures, different number of fuzzy sets will be used for each input variable.

4.5 Conclusions

In this chapter, we have proposed several effective fuzzy model frameworks. The input-output characteristics of the networks are investigated with various numeric connections and selected triangular norms and conorms. The interpretation of the networks is also discussed and pruning mechanism is proposed to allow for some further reduction of the network so as to improve the network interpretability. Based on the proposed logic network, three key technologies are presented here for model development, namely the mechanism of fuzzy equalization, context-based clustering, and the projection and reduction of each variable.

Bibliography

[1] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, New York, Plenum, 1981.

[2] P. Dwinger, Introduction to Boolean algebras, Würzburg, Physica Verlag, 1971.

[3] W. Pedrycz, "Conditional fuzzy c-means", Pattern Recognition Letters, vol. 17, pp. 625-632, 1996.

[4] W. Pedrycz, "Conditional fuzzy clustering in the design of radial basis function neural networks," IEEE Transactions on Neural Networks, vol. 9, pp. 601-612, 1998.

[5] W. Pedrycz, Knowledge-Based Clustering, Wiley, New York, 2005.
[6] W. Pedrycz, "Fuzzy equalization in the construction of fuzzy sets," Fuzzy Sets and Systems, vol. 119, pp. 329-335, 2001.
[7] R.E. Simpson, Introductory Electronics for Scientists and Engineers, 2nd Edition, Boston, MA, Allyn and Bacon, 1987.
[8] S. Susanna, Discrete Mathematics with Applications (2nd Edition), Brooks/Cole Publishing Company, 1995.
[9] J.E. Whitesitt, Boolean Algebra and Its Applications, New York, Dover, 1995.

Chapter 5
Discretization

As shown in Chapter 2, a general fuzzy model consists of three fundamental components, namely input interface, processing core, and output interface. The input interface plays an important role in the information granulation necessary to deal with real-world data. Through the input interface, the processing core of a fuzzy model receives granular information upon which it can perform logic computations. Discretization is a process of aggregation that abstracts real-world numeric data into information granules that are more concise and closer to the knowledge-level representation [5, 20]. Hence discretization is used here as a reasonable way to deal with information granulation. This chapter presents a comprehensive literature review of existing discretization approaches and proposes a new discretization method which is of interest in the context of fuzzy modeling.

5.1. Terms and notations

Before we proceed with describing the process of discretization, it will be useful to clarify some commonly encountered terms and notations.

Variable: also called feature or attribute; refers to an individual measurable property of the data. There are three formats of variables: discrete, continuous, and nominal.

Instance: also called pattern or data point; refers to a collection of feature values. A data set is a collection of instances. Usually a data set is in matrix form where a row represents an instance and a column corresponds to a variable.

Cutoff point: also known as split point; it refers to the value within a certain range of continuous values which divides the range into two intervals, one interval being less than or equal to the cutoff point, the other being greater than the cutoff point.

Discretization level: refers to the number of intervals or partitions of a continuous variable. It is also called discretization region.

Inconsistency: two instances are considered inconsistent (or conflicting) when they are the same in attribute values but different in output value. For example, instance (**a**, 0) and instance (**a**, 1) are considered inconsistent because they have

the same input (**a**) but different outputs (0) and (1).

Least inconsistency: refers to the smallest inconsistency count for a given input. For example, given seven instances with the same input, (**a**, 0), (**a**, 0), (**a**, 0), (**a**, 0), (**a**, 1), (**a**, 1), (**a**, 2). The smallest inconsistency count for the input **a** is 3—[(**a**, 1), (**a**, 1), (**a**, 2)]. Usually the least inconsistency is calculated as the total number of instances with matched input minus the largest count for consistency. In this case, the least inconsistency can be calculated as 7-4=3.

Inconsistency rate: also known as conflict rate, refers to the percentage of the least inconsistent data count over total number of instances. We use ζ to denote the inconsistency rate in this thesis.

Discrete data: values that can be counted. An example of discrete data is the number of cylinders in a car.

Continuous data: all values on the number line within a value range. The value range is usually denoted as [min, max] where min and max are denoted as the lowest and the highest values of the variable range, respectively.

5.2 Approaches to discretization – Literature survey

Real-world data usually comes in a mixed format, such as continuous and discrete, while logic-based processing realized by fuzzy models operates on the more abstract constructs of information granules. Discretization, a bridge between the real-world and a fuzzy model core, plays an important role in fuzzy modeling. There are a large number of discretization approaches available in the literature. In this section, we provide an extensive survey of existing approaches and discuss strengths and weaknesses of each of them.

Equal-width discretization (EWD)

Equal-width discretization (EWD)[2] is one of the simplest methods in the literature to discretize a continuous variable. In particular, it is a binning technique that splits the variable range [min, max] into k intervals of equal width, where k is a user predefined parameter. Each interval is associated with a distinct discrete value. The width of each interval is: w = (max-min)/k, and the cutoff points are min+w, min+2w, …, min+(k-1)w. Figure 5.1 illustrates an idea of such a discretization process:

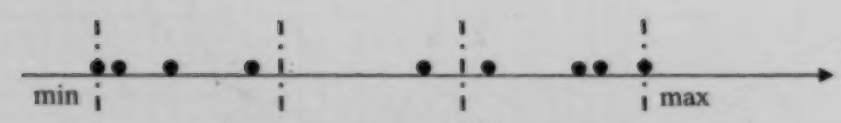


Figure 5.1 Equal-width discretization
(discretize the range of [min, max] into three intervals with same width)

EWD is the most straightforward discretization technique as it does not require any knowledge of the data. But the weakness is obvious as the distribution of cutoff points is sensitive only to the number of intervals that are predefined by the user. Except for the lowest and highest values of the variable, no other output information or domain knowledge is involved during the discretization. This is a typical example of unsupervised discretization.

Equal-frequency discretization (EFD)

Equal-frequency discretization (EFD) is another example of a simple discretization method [2]. It is also known as a binning technique, and the predefined parameter k is used to determine the number of intervals. In EFD, the intervals are created so that each interval contains approximately the same number of continuous variable values. Note that instances with identical values must be placed in the same interval so that it usually results in only an approximate same frequency for each interval.

For instance, if there are N instances for a given variable, then each adjacent interval contains N/k instances. Figure 5.2 shows the mechanism of EFD: nine instances are discretized into three predefined intervals, each of which contains three instances.

Figure 5.2 Equal-frequency discretization
(discretize nine instances into three same frequency intervals)

Compared to EWD, EFD achieves better data understanding because it takes the distribution of data into consideration. However, this method does not consider the output information to determine the discretization cutoff points, thus it is also regarded as an unsupervised discretization technique.

Both EWD and EFD are unsupervised discretization methods and are further

problematic because they can handle the discreitzation of only one variable at a time. Although these weaknesses are well known, both methods receive a lot of usage in data mining and knowledge discovery because of their simplicity and ease of implementation.

As the need for more accurate and efficient discretizations grows, the technology for discretization has developed rapidly. Over the years, many discretization algorithms have been proposed and have demonstrated that discretization has the potential to improve the model and predict accuracy. These methods can be categorized in several dimensions due to different needs: namely: supervised vs. unsupervised[5], splitting (top-down) vs. merging (bottom-up)[13], and univariate vs. multivariate [1].

Supervised vs. Unsupervised Methods that use output class information for the selection of cutoff points during discretization are supervised. Methods that do not use output class information are unsupervised.

Splitting vs. Merging In splitting discretization, initially the whole value range is considered as one interval. The whole value range is then split into sub-intervals until some stopping criteria are met. In merging discretization, intervals initially divided are merged through adjacent intervals until certain stoping criteria are met. Splitting and merging discretizations are often regarded as hierarchical discretizations; splitting discretizations are called top-down methods and merging discretizations are called bottom-up methods.

Univariate vs. Multivariate Methods that discretize one continuous attribute at a time are univariate. Multivariate methods consider multiple attributes and relationships between attributes.

Entropy-based discretization (EBD)

In 1991, Catlett introduced an entropy measure to find potential cutoff points to split a continuous variable [2]. However, the stopping criteria of discretization in his method is rather ad hoc, thus this method did not attract much attention. In 1993, Fayyad and Iraniproposed another entropy-based method that provides a more general way to determine when to stop recursive splitting [7]. The method evaluates as a candidate cutoff point the midpoint between each successive pair of sorted values. For evaluating each candidate cutoff point, the data are discretized into two intervals and the resulting class information entropy is calculated. A minimum description length principle (MDLP) is used to determine when discretization is complete. This is a typical supervised discretization that makes use of output class information when calculating the entropy.

Although EBD has demonstrated an improvement in classification accuracy for naive-Bayesne [4, 14, 25], the method produces a too-coarse granulation of discretization intervals for some attributes caused by applying the MDL principle as a stopping criterion. Also, being a univariate approach EBD has limited available depth [6, 8].

ChiMerge, StatDisc, InfoMerge, and Chi2 discretization

ChiMerge is the first discretization method that applied merging instead of splitting. ChiMerge uses $\chi 2$ statistics to determine if the relative class frequencies of adjacent intervals are distinctly different or if they are similar enough to justify merging them into a single interval [13]. The ChiMerge algorithm is a two-step method. The initial step starts with each instance in an interval. The second step is a bottom-up merging process: compute the $\chi 2$ for each pair of adjacent intervals and merge the pair of adjacent intervals with the lowest $\chi 2$ value. Merging continues until all pairs of intervals have $\chi 2$ values exceeding a $\chi 2$-threshold. ChiMerge requires the $\chi 2$-threshold to be specified manually, ideally one $\chi 2$-threshold for each attribute which is difficult to fulfill. A too big or too small $\chi 2$-threshold will over- or under-discretize an attribute. Thus it is not easy for ChiMerge to find a proper $\chi 2$-threshold for each attribute.

Much work has been done to improve the efficiency and accuracy of ChiMerge discretization. ChiMerge can allow only two intervals to be evaluated and merged at a time. StatDisc discretization, proposed by Richeldi and Rossotto in 1995 [19], extends the ChiMerge method to allow any number of intervals to be merged at a time. StatDisc is still based on a statistical measure to determine the merging and the statistical measures which treat an attribute and a class symmetrically. InfoMerge [10], on the other hand, treats an attribute and a class asymmetrically to discretize the attributes.

Chi2 discretization [16] is another modification to the ChiMerge method. It was intended to overcome the weakness of ChiMerge and let the data itself determine what proper $\chi 2$-threshold should be taken. It uses ChiMerge as a basis of the discretization and enhances the ChiMerge algorithm by introducing inconsistency rate checking as the stopping criterion. When the inconsistency rate is below a predefined level, it indicates that the discretized data set accurately represents the original data set and the discretization stops. However, how is a proper inconsistency rate level to be assigned to different attributes? So far, there have been a few suggestions proposed to solve this problem [22].

All the merging discretization methods discussed here are univariate.

Fuzzy discretization (FD)

Traditional discretization methods divide continuous attributes into a number of intervals whose boundaries are represented by crisp cutoff points. Fuzzy discretization [11], introduced by Ishibuchi *et al.* in 2001, allows overlapping intervals. This method uses domain knowledge to define fuzzy membership functions. The borders of intervals are then represented by membership grades instead of crisp cutoff points. However, the number of intervals, the boundaries of intervals, and the degrees of overlapping are difficult to optimize. Little research has been conducted in this area so far [26].

Cluster-based discretization(CBD)

The methods discussed so far are all univariate discretization. Cluster-based discretization [3], proposed by Chmielewski and Grzymala-Busse in 1996, is the first multivariate discretization method. It consists of two steps. The first step determines the initial intervals based upon the clusters formed in the feature space. The stopping criterion of cluster formation is whether the level of consistency of the partition is less than the level of consistency of the original data. Cutoff points are simultaneously determined in terms of all attributes once the cluster formation is completed. In the second step, the number of discretized intervals is minimized by merging adjacent intervals. Consistency is also checked at this stage to ensure the data consistency after discretization is above the given threshold. The second process stops once all the pairs of adjacent intervals are examined and no further intervals can be merged.

Evolutionary discretization (ED)

Evolutionary discretization was first implemented in EDRL-MD, an evolutionary-algorithm-based system that was built for learning decision rules from the dataset [15]. An evolutionary algorithm (EA) was employed in EDRL-MD. Here each string is composed of n sub-strings where n is the number of attributes and each sub-string encodes a condition related to one attribute. The cutoff points for all continuous attributes are simultaneously formed when the search stops and decision rules are induced. The main advantage of this method is its multivariate discretization ability. However, the fitness function in EA requires some parameters to be chosen on an experimental basis; this reduces the flexibility of the method.

Figure 5.3 shows a systematic hierarchy chart for the discretization methods discussed previously. We divide the methods into two main categories, namely supervised and unsupervised methods; under each category there are two sub-categories, univariate methods and multivariate methods.

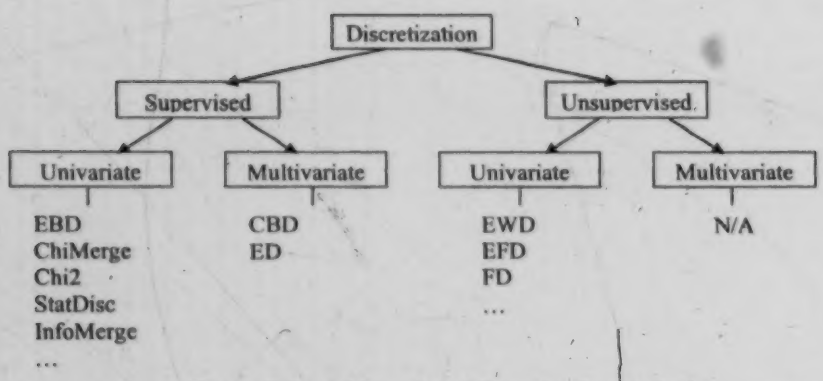


Figure 5.3 A hierarchy of discretization methods

5.3 Proposed modeling environment

In this section we discuss the ways discretization was implemented in our research. Since multivariate discretization is able to capture interdependencies between attributes, and supervised discretization helps select the proper cutoff points by considering output class information, we employ both techniques in our method. Given the different nature of the data and the way in which different variables play in the development of the model, the discretization processes realized for output and input variables are treated differently.

Output variable

For the output variable, we first use the K-Means clustering method [17] with "p" clusters. The center points (prototypes) of these "p" clusters are sorted and arranged in a vector format $\mathbf{m} = [m_1, m_2, \ldots m_{p-1}]$. The "p-1" prototypes give rise to "p" intervals by defining its end points in-between the centers of the clusters. As illustrated in Figure 5.4, the "p-1" cutoff points (mid-point of the successive cluster center, marked as cross) split the range of the output [min, max] into "p" intervals. We denote these "p" intervals by $label_1$, $label_2$, ... $label_p$.

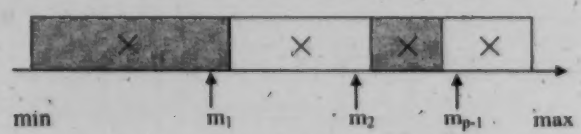


Figure 5.4 Mechanism of discretization by K-Means

Input variable

We assume a "c" discretization level for each input variable which is continuous. The discretization is carried out by choosing cutoff points for all continuous variables simultaneously. With the determination of the cutoff points for input variables, the original dataset is then discretized into discrete datasets.

The objective of discretization processes is to minimize the inconsistency rate for the discretized dataset. In this study we use the "inconsistency rate," which is defined as a percentage of conflicts taken with respect to all data involved in the discretization. More specifically, the inconsistency rate ζ is calculated as follows:

$$\zeta = \frac{\sum_{i=1}^{p} d_i}{N} \qquad i=1,...,p \tag{5.1}$$

where N denotes the number of all instances of the dataset and p is the number of classes for the output discretized by K-Means. $d_i (i = 1, \ldots p)$ is the number of conflicts for each output class.

The discretization process can be summarized as a two-phase processing; the first phase determines the cutoff points for the output by K-Means, and the second phase is to find the optimal cutoff points for the input variables that results in the minimum inconsistency rate. Though many alternatives can be considered to determine the optimal cutoff points of the input variable—equal-width discretization, taking into consideration searching through the space of all features simultaneously, and making use of output class information—particle swarm optimization (PSO) appeals here as a reasonable way to find cutoff points for the input variables.

5.4 Particle swarm optimization (PSO)

Particle swarm optimization (PSO) is an example of social-behavior and population-driven optimization. It was first introduced in 1995 by James Kennedy and Russell C. Eberhart [12]. This technique has attracted much interest and has evolved greatly since then. It has been widely applied in various fields [9, 18, 21, 23, 24, 27].

In PSO, each particle is a possible solution in the multidimensional search space called a problem domain. A particle swarm is a population of particles, that is, a set of possible solutions. Each particle explores the search space and its performance during movement is assessed by means of some performance index

(fitness function). Depending upon the problem at hand, the objective is to either minimize or maximize the given fitness function.

The movement of an individual particle is governed by the two values of the performance index. The first value reported is the best solution achieved by this particle so far. The second value is the value of the best solution obtained so far by all the particles in the population. The position of a particle is described by some vector **Z**(t) where "t" denotes consecutive discrete time moments. Its speed at time "t" is denoted as a vector **V**(t). Given these two values of the performance index, each particle updates its velocity **V** and position **Z** as follows:

$$\mathbf{V}(t+1) = \xi\mathbf{V}(t) + \phi_1(\mathbf{P} - \mathbf{Z}(t)) + \phi_2(\mathbf{P}_{total} - \mathbf{Z}(t)) \quad \text{//update speed (5.2)}$$
$$\mathbf{Z}(t+1) = \mathbf{Z}(t) + \mathbf{V}(t+1) \quad \text{//update position (5.3)}$$

where **P** denotes the best position (characterized by the lowest performance index) reported so far for this particle and $\mathbf{P}_{total}$ is the best position overall developed so far across the whole population. ξ is the inertial weight of [0, 1] which is to articulate some factor of resistance to change the current speed. ϕ_1 and ϕ_2 are random numbers drawn from the uniform distribution formed over the [0, 2] interval, that is U[0, 2]. Figure 5.5 shows the pseudo code of the PSO procedure.

```
For each particle
        Initialize particle
End
Do
  For each particle
     Calculate fitness value
     If the fitness value is better than the best fitness value (pBest) in history
          Set current value as the new pBest
  End
  Select the particle with the best fitness value among all particles as gBest
  For each particle:
     Update particle velocity according to the velocity equation (5.2)
     Apply the velocity constriction
     Update particle position according to the position equation (5.3)
     Apply the position constriction
  End
While maximum iterations or minimum error criteria is not attained
```

Figure 5.5 The pseudo-code of PSO

Relating the PSO to the problem of discretization, the search space comprises all cutoff points for all input variables to be discretized. In other words, each particle **Z** is encoded by all cutoff points of input variables. The dimensionality of each particle **Z** is n(c-1) in length given that we are concerned with "n" variables with the discretization regions of "c." And the minimized performance index of each particle is evaluated by the inconsistency rate based on the discretization given by the particle. The number of discretization regions "c" are predetermined and are not a part of the PSO optimization process.

Once the discretization has been completed, the continuous dataset is transformed into a discrete dataset. But such discrete datasets usually contain inconsistent data which will in turn affect the accuracy of the modeling. In this sense, eliminating conflict data becomes an essential aspect after the discretization. The simplest way of eliminating conflicts is by removing inconsistent data from the overall data set. Such resulting datasets are called clean datasets. By applying the one-out-of-n strategy to each variable, as well as the output of the clean dataset, it becomes a collection of Boolean input – output pairs in the form of

$$\{ \mathbf{x}_1(k) \quad \mathbf{x}_2(k) \ldots \mathbf{x}_n(k) \quad \tilde{\mathbf{y}}(k) \} \quad k=1,2,\ldots,N$$

where $\mathbf{x}_i(k) \in \{0,1\}^c$ if the original i-th variable was continuous and $\mathbf{x}_i(k) \in \{0,1\}^{c'}$ with c'<=c in the case of the discrete variable. Furthermore, $\tilde{\mathbf{y}} \in \{0,1\}^p$.

5.5 Illustrative example

In this section, we use one synthetic data set to illustrate the discretization methods we have discussed previously. We compare the results by using EWD, EFD, EBD, ChiMerge, Chi2, and our proposed method (PSO+K-Means).

Consider a three-input-one-output dataset, generated from the following equation:

$$y(k) = \max(x_1(k), \min(x_2(k), x_3(k))) \qquad k=1, 2, \ldots, 100$$

x_1, x_2, and x_3 are all random variables with values in the range of [0, 1]. y is classified into three classes with the following equation:

$$y = \begin{cases} 1 & y < 0.4 \\ 2 & 0.4 \le y \le 0.7 \\ 3 & y > 0.7 \end{cases}$$

We summarize the results in Table 5.1, which illustrates the cutoff points and inconsistency rate of each method.

Table 5.1 Discretization results for the synthetic data

Method	Distribution of cutoff points	Inconsistency rate
EWD	x1: [1/3, 2/3] x2: [1/3, 2/3] x3: [1/3, 2/3]	21%
EFD	x1: [0.092, 0.328] x2: [0.248, 0.512] x3: [0.377, 0.733]	6%
EBD	x1: [0.145, 0.257] x2: [0.242, 0.631] x3: [0.327, 0.728]	8%
ChiMerge	x1: [0.079, 0.183, 0.214, 0.358, 0.603, 0.813, 0.887, 0.931] x2: [0.034, 0.211,0.267] x3: [0.243, 0.355, 0.413, 0.706]	5%
Chi2	x1: [0.092, 0.243, 0.509, 0.757, 0. 869, 0.927] x2: [0.194, 0.509,0.771,0.857, 0.921] x3: [0.292, 0.536, 0.663, 0.864]	3%
PSO+K-Means	x1: [0.167, 0.473] x2: [0.294, 0.656] x3: [0.325, 0.791]	1%

Table 5.1 shows the different distributions of cutoff points obtained by different methods. Among them, ChiMerge and Chi2 need the most cutoff points. This is because of the bottom-up nature of the ChiMerge and Chi2 methods. Of all the methods considered, PSO+K-Means has the lowest inconsistency rate (1%) and EWD has the highest inconsistency rate (21%).

5.6 Conclusions

We have conducted an extensive literature review of the existing discretization methods and summarized these methods in a hierarchy chart. These methods are divided into two main categories, supervised and unsupervised discretization.

Unsupervised discretization is widely used due to its simplicity and ease of implementation. Supervised discretization makes use of output class information, which in turn helps improve the discretization results. Univariate vs. multivariate and splitting vs. merging are considered as sub-categories of these two. The modeling environment has been defined in terms of K-Means and particle swarm optimization (PSO) in this research. In particular, we discretize the output by K-Means clustering, then with the help of PSO we discrete all of the features simultaneously. The illustrative example shows promising results for such a modeling environment. The originality lies in the comparative analysis between the proposed discretization method and some existing discretization methods in literature.

Bibliography

[1] S.D. Bay, "Multivariate discretization of continuous variables for set mining", Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 315-319, 2000.

[2] J. Catlett, "On changing continuous attributes into ordered discrete attributes", Proceedings of the Fifth EuropeanWorking Session on Learning, pp. 164-177, 1991.

[3] M.R. Chmielewski, and J.W. Grzymala-Busse, "Global discretization of continuous attributes as preprocessing for machine learning", International Journal of Approximate Reasoning, vol. 15, pp. 319-331, 1996.

[4] B. Domingos, and M. Pazzani, "Beyond independence: conditions for the optimality of the simple Bayesian classifier", Proceedings of Thirteenth International Conference on Machine Learning, pp. 105-112, 1996.

[5] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features", Proceedings of the Twelfth International Conference on Machine Learning, pp. 194-202, 1995.

[6] U.M. Fayyad, and K.B. Irani, "On the handling of continuous-valued attributes in decision tree generation", Machine Learning, vol. 8, pp. 87-102, 1992.

[7] U.M. Fayyad, and K.B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning", Proceedings of the 13th International Joint Conference on Arrijicial Intelligence, pp. 1022-1027,1993.

[8] U.M. Fayyad, and K.B. Irani, "Discretizing continuous attributes while learning bayesian networks", Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufmann, pp. 157-165, 1996.

[9] N. Franken, and A.P. Engelbrecht, "Comparing particle swarm optimisation structures to learn the game of checkers from zero knowledge", Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), pp. 234-241, 2003.

[10] A.A. Freitas, and S.H. Lavington, "Speeding up knowledge discovery in

large relational databases by means of a new discretization algorithm", In Advances in Databases, Proceedings of the 14th British National Conference on Databases, pp. 124-133, 1996.
[11] H. Ishibuchi, T. Yamamoto, and T. Nakashima, "Fuzzy data mining: effect of fuzzy discretization", The 2001 IEEE International Conference on Data Mining, 2001.
[12] J. Kennedy, and R.C. Eberhart, "Particle swarm optimization", Proceedings IEEE International Conference Neural Networks, IEEE Press, Piscataway, New Jersey, vol. 4, pp. 1942-1948, 1995.
[13] R. Kerber, "Chimerge: Discretization for numeric attributes", In National Conference on Artificial Intelligence, pp. 123-128, 1992.
[14] P. Kontkaren, P. Myllymaki, T. Silander, and H. Tirri, "Bayda: Software for bayesian classification and feature selection", The 4th International Conference on Knowledge Discovery and Data Mining, pp. 254-258, 1998.
[15] W. Kwedlo, and M. Kretowski, "An evolutionary algorithm using multivariate discretization for decision rule induction", Principles of Data Mining and Knowledge Discovery - Lecture Notes in Artificial Intelligence, vol. 1704, pp. 392-397, 1999.
[16] H. Liu, and R. Setiono, "Feature selection and discretization", IEEE Transactions on Knowledge and Data Engineering, vol. 9, pp. 1-4, 1997.
[17] J.B. MacQueen, "Some methods for classification and analysis of multivariate observations", Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281-297, 1967.
[18] S. Paterlini, and T. Krink, "Differential evolution and particle swarm optimisation in partitional clustering", Computational Statistics Data Analysis, vol. 50, no. 1, pp. 1220-1247, 2006.
[19] M. Richeldi, and M. Rossotto, "Class-driven statistical discretization of continuous attributes (extended abstract)", In European Conference on Machine Learning, pp. 335-338, 1995.
[20] H.A. Simon, The Sciences of The Artificial, 2nd Edition, Cambridge, Massachusetts, MIT Press, 1981.
[21] T. Sousa, A. Neves, and A. Silva, "Swarm optimization as a new tool for data mining", Proceedings of the Parallel and Distributed Processing Symposium, pp. 144-149, 2003.
[22] E.H. Tay, and L. Shen, "A modified chi2 algorithm for discretization", IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 3, pp. 666-670, May/June 2002.
[23] J.C. Tillett, R.M. Rao, F. Sahin, and T.M. Rao, "Particle swarm optimization for the clustering of wireless sensors", Proceedings of SPIE, Digital Wireless Communications V, vol. 5100, pp. 73-83, 2003.
[24] G. Venter, and J. Sobieszczanski-Sobieski, "Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization", Structural and Multidisciplinary Optimization, 2004.
[25] X. Wu, "A bayesian discretizer for real-valued attributes", The Computer

Journal, vol. 39, no. 8, pp. 688-691, 1996.
[26] X. Wu, "Fuzzy interpretation of discretized intervals", IEEE Transactions on Fuzzy Systems, vol. 7, no. 6, pp. 753-759, 1999.
[27] W. Zhang, Y. Liu, and M. Clerc, "An adaptive PSO algorithm for reactive power optimization", Advances in Power System Control Operation and Management (APSCOM2003), Hongkong, 2003.

Chapter 6
Learning

Network learning is a critical phase during network development. Learning usually includes structural optimization and parametric optimization. Structural optimization seeks the optimal structure and parametric optimization tunes the required parameters based on the optimal structure. In this chapter we first assess the overall development of the logic network. Based upon the assessment of the development environment, we then proceed with the detailed learning of the networks. Because the structure of the network is usually predetermined, the discussion of learning in this chapter will be primarily concerned with supervised parametric learning. Parametric learning aims to optimize the parameters of the fuzzy neural network and can be accomplished in many different ways [2-7]. Among these approaches, we focus on gradient-based learning, particle swarm optimization, and a hybrid of these two approaches.

6.1 Assessment of the overall development of a logic network

In the fuzzy neural network shown in Figure 6.1 there are three important layers, namely input, hidden, and output. In this construct the neurons are fully connected between each layer. The role of the connections of fuzzy neurons is to weigh the inputs and in this way offer them the required parametric flexibility. Several groups of parameters contribute to this flexibility; three groups are discussed below.

1. The number of neurons in the hidden layer.

While the number of inputs (n) and outputs (m) are predetermined by the specific problem itself, the number of the neurons in the hidden layer can be freely selected. The choice of the number of neurons in the hidden layer directly impacts the topology of the network and will very likely affect the accuracy of the mapping realized by the network. We anticipate that increasing the number of neurons will result in increased accuracy of the input-output mapping. However, a high number of neurons will cause an associated memorization effect, that is, as the number of neurons becomes high, we start losing the generalization ability of the network.

2. The choice of t- norms and conorms (s-norms) in the realization of neurons.

With triangular norms and co-norms we are faced with an enormous diversity. The neurons can be realized with the use of different combinations of t- norms and s-norms. We consider several pairs of t-norms and co-norms such as (min, max), (product, probabilistic sum), (Lukasiewicz *and*, Lukasiewicz *or*). These pairs of t- and s-norms are very typical and have been used in a number of studies. Although other pairs are available, we do not envision a substantial gain from a more thorough exploration of other alternatives for this project.

3. Numeric values of the connections between neurons, and the identity values of unineurons.

The connectives of neurons are essential parameters that can offer superb learning ability to the logic-network. By changing the numeric values of these connectives, we can achieve flexibility in the network.

Optimization of the network parameters is an essential part of the overall development process. The third group of parameters (connectives and identity values) is a focal point during learning efforts. Their values offer the unique possibility of optimizing the network. Our learning efforts will concentrate on them.

Regardless of the implementation of the network, the principle idea of supervised parametric learning can be generalized as follows. Given a fuzzy neural network as shown in Figure 6.1, connectives between the input layer and the hidden layer are denoted by **v**, connectives between the hidden layer and the output layer are denoted by **w**. The objective of network learning is to modify the parameters **v** and **w** so that the difference between the output (**y**) and the target output (Target) can be minimized.

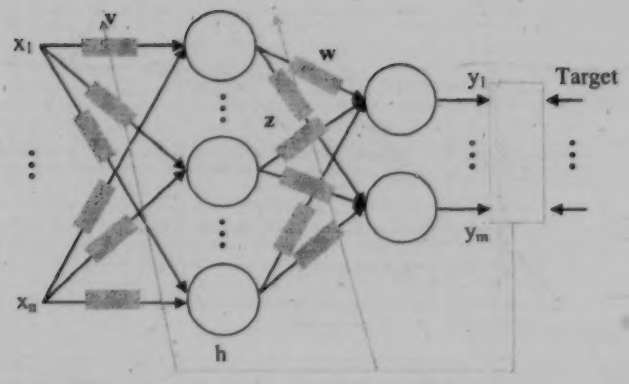


Figure 6.1 The optimization of network parameters v and w

6.2 Internal and external performance index of the network

A typical way to measure the difference between the desired (target) output and the output values of the network (model) is by calculating the sum of the squared errors (differences between model output and target output). Such a process provides an internal performance index Q, defined as follows:

$$Q = \frac{1}{N}\sum_{k=1}^{N}\|\mathbf{y}(k) - \mathbf{t}(k)\|^2 \qquad (6.1)$$

where $\mathbf{y}(k)$ corresponds to the k-th data point of model output, $\mathbf{t}(k)$ is denoted as the corresponding target output, and $\|\cdot\|$ is the Euclidean distance.

To assess the performance of the model with respect to the external world, we need to decode the results from the "internal" world of the unit hypercube and translate it back to the original output space. The decoding of $\mathbf{y}$, gives rise to a numeric entity $u \in \mathbf{R}$, expressed in the form:

$$u = \frac{\sum_{r=1}^{m} c_r y_r}{\sum_{r=1}^{m} y_r} \qquad (6.2)$$

where c_r (r=1,…,m) are entries of the center vector $\mathbf{c} = [c_1, c_2 \ldots, c_m]$.

The external performance index V is a root mean squared error (RMSE) of deviations between the model and the data, that is:

$$V = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(u_k - target_k)^2} \qquad (6.3)$$

where N denotes the size of a training or testing data set, u_k stands for the decoded result of y_k which is produced by the model that corresponds to the k-th data point, and $target_k$ is the original k-th data point of the output.

Note that the use of the performance index V applies to problems involving continuous outputs. In the case of classification problems, we can still consider the decoding procedure, however, the performance of the model will be expressed in terms of the classification error.

6.3 Gradient-based learning

Gradient-based learning has been successfully applied to the training of many applications [1, 8, 9]. Denoting all connections of the network by **conn**, the general scheme of gradient-based learning can be qualitatively written as:

$$\mathbf{conn}(\text{iter}+1)=\mathbf{conn}(\text{iter})- \alpha\nabla_{\mathbf{conn}} Q \qquad (6.4)$$

where α is a positive learning rate and Q is the internal performance index expressed in equation (6.1).

If we confine ourselves to the product (t-norm) and probabilistic sum (t-conorm), we can proceed with detailed computations of the gradient standing in the updates of the connections. Expressions for the AND-OR neurons can then be written as follows:

$$\begin{cases} z_{js} = \prod_{i=1}^{n}(x_i + v_{ij} - x_i v_{ij}) & j = 1, \cdots, h \qquad (6.5) \\ y_s = 1 - \prod_{j=1}^{h}(1 - z_{js} w_{js}) & s=1, \ldots, m \qquad (6.6) \end{cases}$$

In the sequel we obtain

$$\mathbf{v}(\text{iter}+1) = \mathbf{v}(\text{iter}) - \alpha\nabla_{\mathbf{v}} Q \qquad (6.7)$$

$$\mathbf{w}(\text{iter}+1) = \mathbf{w}(\text{iter}) - \alpha\nabla_{\mathbf{w}} Q \qquad (6.8)$$

where

$$\frac{\partial Q}{\partial w_{js}} = \frac{\partial Q}{\partial y_s}\frac{\partial y_s}{\partial w_{js}} = \frac{2}{N}\sum_{k=1}^{N}(y_s(k)-t_s(k))\frac{\partial y_s}{\partial w_{js}} = \frac{2}{N}\sum_{k=1}^{N}(y_s(k)-t_s(k))[z_{js}\prod_{\substack{r=1\\ r\neq j}}^{h}(1-z_{rs}w_{rs})]$$

$s=1....m;\quad j=1....h$ (6.9)

$$\frac{\partial Q}{\partial v_{ij}} = \sum_{s=1}^{m}\frac{\partial Q}{\partial y_s}\frac{\partial y_s}{\partial z_{js}}\frac{\partial z_{js}}{\partial v_{ij}} = \frac{2}{N}\sum_{s=1}^{m}\sum_{k=1}^{N}(y_s(k)-t_s(k))[w_{js}\prod_{\substack{r=1\\ r\neq j}}^{h}(1-z_{rs}w_{rs})][\prod_{\substack{r=1\\ r\neq i}}^{n}(x_r+v_{rj}-x_r v_{rj})(1-x_i)]$$

$i=1.....n$ (6.10)

If the network is constructed with the aid of unineurons, the network can be expressed as follows:

$$z_{js} = \overset{n}{\underset{i=1}{T}}(u(x_i, v_{ij}, g_{ij})) \qquad i=1,\cdots,n;\ j=1,\cdots,h$$

$$= \prod_{i=1}^{n}\begin{cases} g_{ij}\dfrac{x_i}{g_{ij}}\dfrac{v_{ij}}{g_{ij}} & x_i, v_{ij}\in[0,g_{ij}] \\ g_{ij}+(x_i-g_{ij})+(v_{ij}-g_{ij})-\dfrac{(x_i-g_{ij})(v_{ij}-g_{ij})}{1-g_{ij}} & x_i, v_{ij}\in[g_{ij},1] \\ \max(x_i, v_{ij}) & \text{otherwise} \end{cases}$$

(6.11)

$$y_s = \overset{h}{\underset{j=1}{S}}(u(z_{js}, w_{js}, q_{js})) \qquad j=1,\cdots,h \qquad s=1,\cdots,m$$

$$= \begin{cases} 1-\prod_{j=1}^{h}[1-q_{js}\dfrac{w_{js}}{q_{js}}\dfrac{z_{js}}{q_{js}}] & z_{js}, w_{js}\in[0,q_{js}] \\ 1-\prod_{j=1}^{h}[1-q_{js}-(z_{js}-q_{js})-(w_{js}-q_{js})+\dfrac{(z_{js}-q_{js})(w_{js}-q_{js})}{1-q_{js}}] & z_{js}, w_{js}\in[q_{js},1] \\ 1-\prod_{j=1}^{h}[1-\max(z_{js}, w_{js})] & \text{otherwise} \end{cases}$$

(6.12)

In the second step, the gradient-based learning for the network is:

$$v(\text{iter}+1) = v(\text{iter}) - \alpha\nabla_v Q \qquad (6.13)$$

$$w(\text{iter}+1) = w(\text{iter}) - \alpha\nabla_w Q \qquad (6.14)$$

Where

$$\nabla_w Q = \frac{\partial Q}{\partial w_{js}} = \frac{2}{N}\sum_{k=1}^{N}(y_s(k) - t_s(k))\frac{\partial y_s}{\partial w_{js}} = \frac{2}{N}\sum_{k=1}^{N}(y_s(k) - t_s(k))$$

$$\begin{cases} \dfrac{z_{js}}{q_{js}}\displaystyle\prod_{\substack{r=1\\ r\neq j}}^{h}\left(1-\frac{w_{rs}z_{rs}}{q_{rs}}\right) & z_{js}, w_{js} \in [0, q_{js}] \\ \left(\dfrac{z_{js}-q_{js}}{1-q_{js}} - 1\right)\displaystyle\prod_{\substack{r=1\\ r\neq j}}^{h}\left[1 - z_{rs} - w_{rs} + q_{rs} + \frac{(z_{rs}-q_{rs})(w_{rs}-q_{rs})}{1-q_{rs}}\right] & z_{js}, w_{js} \in [q_{js}, 1] \\ -\dfrac{\partial}{\partial w_{js}}\displaystyle\prod_{j=1}^{h}[1 - \max(z_{js}, w_{js})] & \text{otherwise} \end{cases} \tag{6.15}$$

For the derivative in the last condition in equation (6.15),

$$-\frac{\partial}{\partial w_{js}}\prod_{j=1}^{h}[1 - \max(z_{js}, w_{js})] = \begin{cases} \displaystyle\prod_{\substack{r=1\\ r\neq j}}^{h}(1 - w_{rs}) & z_{js} \le w_{js} \\ 0 & z_{js} > w_{js} \end{cases} \tag{6.16}$$

s=1, 2, ...,m

$$\nabla_v Q = \sum_{s=1}^{m}\frac{\partial Q}{\partial y_s}\frac{\partial y_s}{\partial z_{js}}\frac{\partial z_{js}}{\partial v_{ij}} = \frac{2}{N}\sum_{s=1}^{m}\sum_{k=1}^{N}(y_s(k) - t_s(k))\frac{\partial y_s}{\partial z_{js}}\frac{\partial z_{js}}{\partial v_{ij}}$$

where

$$\frac{\partial y_s}{\partial z_{js}} = \begin{cases} \dfrac{w_{js}}{q_{js}}\displaystyle\prod_{\substack{r=1\\ r\neq j}}^{h}\left(1-\frac{w_{rs}z_{rs}}{q_{rs}}\right) & z_{js}, w_{js} \in [0, q_{js}] \\ \left(\dfrac{w_{js}-q_{js}}{1-q_{js}} - 1\right)\displaystyle\prod_{\substack{r=1\\ r\neq j}}^{h}\left[1 - z_{rs} - w_{rs} + q_{rs} + \frac{(z_{rs}-q_{rs})(w_{rs}-q_{rs})}{1-q_{rs}}\right] & z_{js}, w_{js} \in [q_{js}, 1] \\ -\dfrac{\partial}{\partial z_{js}}\displaystyle\prod_{j=1}^{h}[1 - \max(z_{js}, w_{js})] & \text{otherwise} \end{cases} \tag{6.17}$$

For the derivative in the last condition in equation (6.17),

$$-\frac{\partial}{\partial z_{js}}\prod_{j=1}^{h}[1 - \max(z_{js}, w_{js})] = \begin{cases} 0 & z_{js} \le w_{js} \\ \displaystyle\prod_{\substack{r=1\\ r\neq j}}^{h}(1 - z_{rs}) & z_{js} > w_{js} \end{cases} \tag{6.18}$$

$$\frac{\partial z_{js}}{\partial v_{ij}} = \begin{cases} \dfrac{x_i}{g_{ij}} \prod_{\substack{r=1 \\ r \neq j}}^{h} \dfrac{x_r v_{rj}}{g_{rj}} & x_i, v_{ij} \in [0, g_{ij}] \\ (1 - \dfrac{x_i - g_{ij}}{1 - g_{ij}}) \prod_{\substack{r=1 \\ r \neq j}}^{h} [x_r + v_{rj} - g_{rj} - \dfrac{(x_r - g_{rj})(v_{rj} - g_{rj})}{1 - g_{rj}}] & x_i, v_{ij} \in [g_{ij}, 1] \\ \dfrac{\partial}{\partial v_{ij}} [\max(x_i, v_{ij})] & \text{otherwise} \end{cases} \tag{6.19}$$

For the derivative in the last condition in equation(6.19),

$$\frac{\partial}{\partial v_{ij}} [\max(x_i, v_{ij})] = \begin{cases} 1 & x_i \leq v_{ij} \\ 0 & x_i > v_{ij} \end{cases} \tag{6.20}$$

Quite often, it is possible to include a momentum term during the calculation of $\nabla_{conn} Q$. The update of the connections is governed by

$$\textbf{conn}(\text{iter}+1) = \textbf{conn}(\text{iter}) - \alpha \nabla_{conn} Q + \beta \nabla_{conn} Q \tag{6.21}$$

where β is the momentum term.

Because of the logic operations of neurons (and unineurons), the connections of neurons (and unineurons) must be confined to the unit interval during and after the gradient-based optimization. The learning rate is usually set up between 0 and 1. The learning may become very slow if the learning rate is very close to 0. However, learning will easily get stuck in a local minimum of the performance index if the learning rate is set too high, say very close to 1.

6.4 Learning with the use of PSO

When dealing with a highly nonlinear character of some dependencies and a high dimensionality of search space, gradient-based learning can easily be compromised. In anticipation of this it is helpful to investigate other options offered by biologically inspired optimization. Particle swarm optimization (PSO) is a reasonable way to support the optimization of the logic networks. To illustrate the benefits of such a composition, three learning strategies will be envisioned with different parameters.

6.4.1 Strategy-1 PSO for all parameters of the network that are binary {0, 1}

The logic network is trained by the binary PSO which considers the entry of each particle as having a value of either 0 or 1, see Figure 6.2. In particular, the network becomes a Boolean network and the output of the network has a Boolean output. The organization of a particle follows the structure of the network, starting with the connections **v** and **w**. If the network is built by unineurons, then the identity values **g** and **q** are sequentially concatenated at the end.

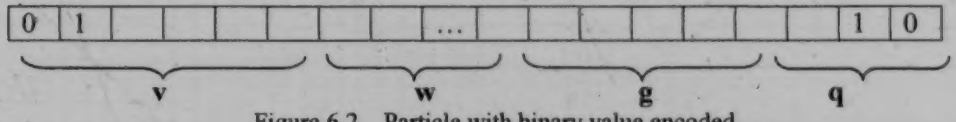


Figure 6.2 Particle with binary value encoded

6.4.2 Strategy-2 PSO for all parameters that are in the range of [0, 1]

In this learning scenario, the connections of the network are all the unit interval values [0, 1]. Each particle in the PSO is encoded as a string with all connections concatenated in the same manner as Strategy-1.

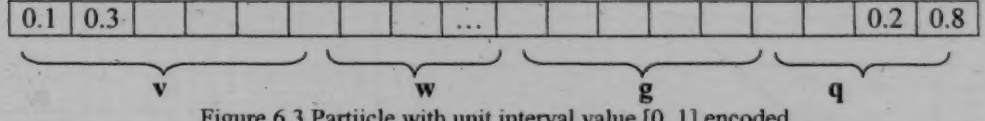


Figure 6.3 Partiicle with unit interval value [0, 1] encoded

6.4.3 Strategy-3 Hybrid of PSO and gradient-based learning

This strategy takes advantage of PSO and gradient-based learning by hybridizing

them. In particular, the identity values **g** and **q** are trained by PSO and the gradient-based learning takes the values of **g** and **q**, and refines the connections **v** and **w** guided by an internal performance index. Each particle has some gradient-based learning which works individually to calculate the performance index and return a value back to the particle as the fitness value in PSO. The overall process can be regarded as two-level learning, with PSO working at the first level and gradient-based learning nested in the second level. The detailed scheme of this hybrid learning is summarized in Figure 6.3.

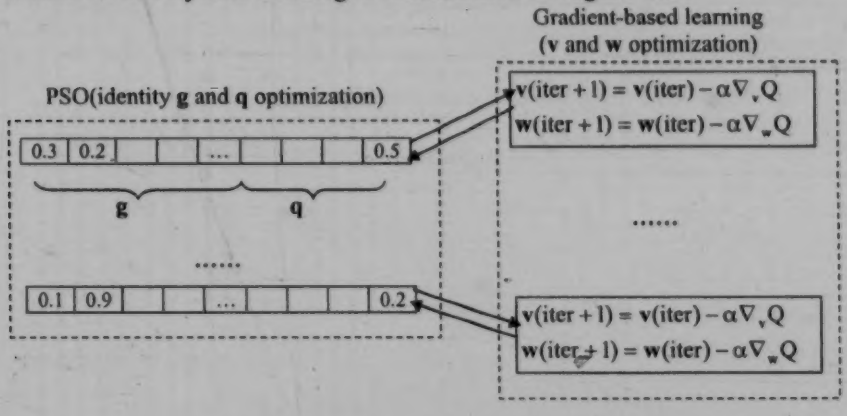


Figure 6.3 Hybrid learning scheme
gradient-based learning as nested learning inside the PSO

Note that Strategy-3 is suitable only for a logic network constructed with unineurons.

In Strategy-1 and Strategy-2 the PSO technique is used to optimize the network parameters. The difference is that Strategy-1 considers only the binary connections of the network, whereas the connections trained by Strategy-2 have a continuous value [0, 1]. However, the drawback of these two strategies is obvious. With an increase in optimized parameters, the length of the particles in PSO expands accordingly. For instance, for a given n-input-m-output unineuron network with "h" hidden AND_U and the length of the particle is 2*h*(n+m); when the h becomes h+1 the increase of length is 2*(n+m). Thus, the computation of learning becomes more challenging. In addition, there are too many parameters encoded in a single particle; that is, too many parameters need to be adjusted at one time, so that efficiency of the learning is easily compromised when the network becomes large or inputs multiply. In Strategy-3, parameter optimization is split into two levels, with each level optimizing some but not all of the parameters. The identity value optimization is carried out at the top level, while the connection optimization is performed at a lower level which acts like nested learning inside the top level. This approach makes the identity optimization disjointed from the phase that is concentrated on connection

adjustments. In this case, the PSO searches the space of all possible identity values, and passes the identity values into the gradient-based learning for further exploration of the connections. The performance returned by gradient-based learning in turn acts as the fitness value which guides the PSO at the top level for further optimization. However, the computation cost of this hybrid learning strategy is the highest among the three strategies.

6.5 Fuzzy partition

The model constructed so far has been formed with the use of Boolean partition. To construct the fuzzy model, we refine the interval form of the information.

Output space. In the case of continuous output, we form a family of triangular fuzzy sets spanning over the intervals we have formed so far. The modal values of these fuzzy sets are taken as the centers of the intervals. Overlap between neighboring fuzzy sets is up to ½. The choice of this form of fuzzy sets is motivated by their lossless reconstruction capabilities. For discrete outputs (characterizing classification problems), no transformation is required.

Input space. Here we construct trapezoidal fuzzy sets over the intervals by retaining some portion of the interval whose length is specified by the core [0, 1], while the rest of the characteristic function is transformed into the linearly increasing and decreasing sections of the membership function. The ½ overlap between consecutive fuzzy sets is retained. The essence of this construct is illustrated in Figure 6.4.

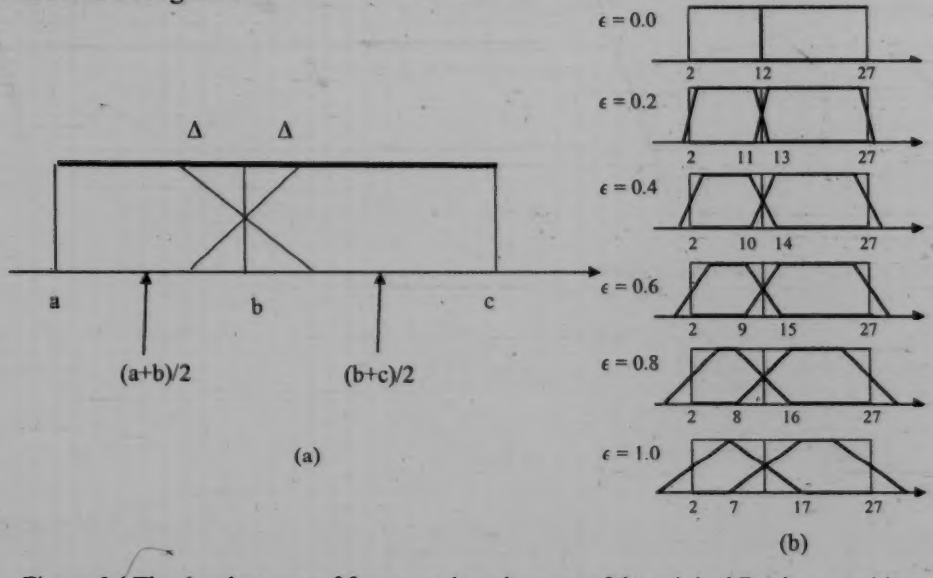


Figure 6.4 The development of fuzzy sets based on use of the original Boolean partition

The use of the core parameter ε in [0, 1] controls the region of transition viz. the argument when the membership grades assume values between 0 and 1. Note that the value of Δ is determined as follows (see Figure 6.4 (a)): $\Delta = \epsilon \min(\frac{c-b}{2}, \frac{b-a}{2})$. In the case $\varepsilon = 0$, we retain the original Boolean partition. Here an increase in the value of ε leads to a more profound visibility of the fuzzy sets. The given value of ε converts to trapezoidal fuzzy sets; these fuzzy sets need not be symmetric. Figure 6.4 (b), illustrates the formation of fuzzy sets with different values of ε. When $\varepsilon = 0$, both the left and right partitions are Boolean; when ε is between 0 and 1, the fuzzy sets for left and right partitions are both trapezoidal fuzzy sets; when $\varepsilon = 1$, the fuzzy set for the left partition becomes a triangular fuzzy set, whereas the fuzzy set for the right partition remains trapezoidal.

6.6. Conclusions

In this chapter, we assessed the network parameters that contribute to the flexibility of logic networks. These parameters include the number of neurons in the hidden layer, the combination of selected t- and s-norms, and the values of the connections. Based on the assessment, we discussed approaches to parametric learning. Two performance indices are investigated during the discussion. We also introduced the core parameter to construct fuzzy partitions that further examined the performance of the logic network.

Bibliography

[1] F. Diotalevi, M. Valle, and D.D. Caviglia, "Evaluation of gradient descent learning algorithms with an adaptive local rate technique for hierarchical feed forward architectures", International Joint Conference on Neural Networks, vol. 2, pp. 185-190, 2000

[2] X. Liang, and W. Pedrycz, "Fuzzy logic-based networks: A study in logic data interpretation", International Journal of Intelligence Systems, vol. 21, no. 12, pp. 1249-1267, 2006.

[3] W. Pedrycz, "Heterogeneous fuzzy logic networks: Fundamentals and development studies", IEEE Transactions on Neural Networks, vol. 15, pp. 1466-1481, 2004.

[4] W. Pedrycz, and F. Gomide, An Introduction to Fuzzy Sets: Analysis and Design, MIT Press, Cambridge, MA, 1998.

[5] W. Pedrycz, and K. Hirota, "Uninorm-based logic neurons as adaptive and interpretable processing constructs", Soft Computing, vol. 11, no. 1, pp. 41-52,

2007
[6] W. Pedrycz, M. Reformat, and C. W. Han, "Cascade architectures of fuzzy neural networks", Fuzzy Optimization and Decision Making, vol. 3, no. 1, pp. 5-37, 2004.
[7] W. Pedrycz, M. Reformat, and K. Li, "OR/AND neurons and the development of interpretable logic models", IEEE Transactions on Neural Networks, vol. 17, no. 3. pp. 636-658, 2006.
[8] N. Qian, "On the momentum term in gradient descent learning algorithms", Neural Networks, vol. 12, no. 1, pp. 145-151, 1999
[9] F. Rosati, P. Campolucci, and F. Piazza, "A general approach to gradient based learning in multirate systems and neural networks", International Joint Conference on Neural Networks, pp. 569-576, 2000

Chapter 7
Experimental studies

In this chapter, we report on a number of experiments carried out for selected machine learning datasets [http://archive.ics.uci.edu/ml/] and datasets from other resources [1-2]. We consider two types of topology of fuzzy neural networks discussed in the previous chapters, namely the networks constructed by AND and OR neurons, and the networks built by AND and OR unineurons. In the former, we apply the fuzzy equalization and conditional fuzzy c-means algorithm for the information granulation. After the learning of the network, we examine the interpretation of the model by means of the pruning mechanism. In the latter, discretization methods (particle swarm optimization and K-Means) are applied at the interface of information granulation, and the three learning strategies discussed in chapter 6 are employed for network training. In addition, both Boolean data and fuzzy data are examined with regard to their relevance in network performance.

7.1 Networks constructed by AND and OR neurons

In this section, the obtained results are presented in a uniform manner by quantifying the approximation abilities of the corresponding models and showing the details of the resulting logic description of the data. We also point at some tradeoffs between the accuracy of the logic models and their interpretability. Throughout the experiments, we used 60% of the data for the training; the remaining 40% is used for the testing.

7.1.1 Boston housing data

This dataset concerns a description of real estate in the Boston area and its price. Each real estate is characterized by a number of features such as

1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. NOX: nitric oxides concentration (parts per 10 million)
5. RM: average number of rooms per dwelling
6. AGE: proportion of owner-occupied units built prior to 1940
7. DIS: weighted distances to five Boston employment centres

8. RAD: index of accessibility to radial highways
9. TAX: full-value property-tax rate per $10,000
10. PTRATIO: pupil-teacher ratio by town
11. B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
12. LSTAT: % lower status of the population

The output (MEDV) is a median value of the home expressed in $ 1000s. Following the overall development scheme intriduced in this study, we start with the fuzzy equalization completed in the output space. The meaningful fuzzy sets we could define there quantify the values of the house as LOW, MEDIUM and HIGH. These three terms are semantically sound and offer enough discrimination. The histogram of the output shown in Figure 7.1 is quite symmetrical with an exception of an elongated tail of higher values of the real estate. We note however that these values occur quite seldom and could be removed from the construction of the fuzzy sets.

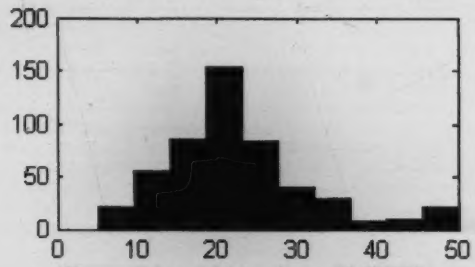


Figure 7.1 A histogram of MEDV

We distribute the fuzzy sets in the space by eliminating all data points that are more than 2σ distant from the mean value of the population of all data; note that we make this requirement stronger than the standard one encountered in statistics that uses a 3σ rule. By completing the fuzzy equalization, we end up with the three linguistic labels ($p = 3$) for the output as shown in Figure 7.2.

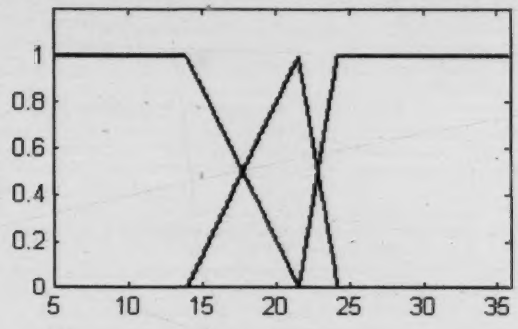


Figure 7.2 Fuzzy sets constructed in the output space
(being the result of outlier elimination and fuzzy equalization)

Then context-based clustering is applied, followed by the projection on the individual input variables and possible reduction (merging) of the adjacent fuzzy sets. Consider the case of 3 clusters(c=3) for each context. Figure 7.3 illustrates the number of fuzzy sets produced for each input variable with respect to different values of ε used in the merging criterion. Without any merging (ε=0), we have c*p+2 =11 fuzzy sets for each variable. For higher values of ε, this number of fuzzy sets starts decreasing while the rate of decrease depends on the specific variable. Noticeably the differences are quite visible ranging between 2 and 5 linguistic terms.

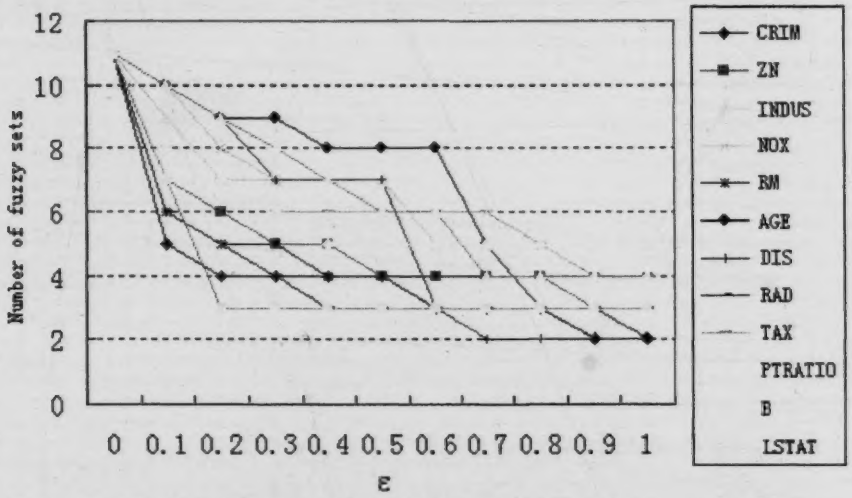


Figure 7.3 Number of fuzzy sets for each input variable with respect to different

Figure 7.4 shows the overall number of fuzzy sets (as being counted for all variables) for the inputs with 3, 5, 9 clusters per each context when being plotted versus the varying values of ε

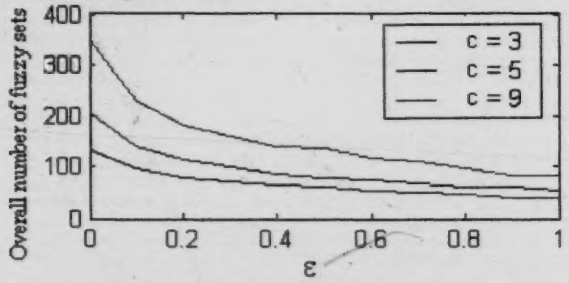


Figure 7.4 Overall number of fuzzy sets for the inputs with 3,5 and 9 clusters

By arbitrarily choosing 3 clusters for each context along with ε =0.8 (at which value we obtain a relative small number of fuzzy sets), see Figure 8, we formed the following linguistic terms for each input variable

1. CRIM = {LOW, MEDIUM, HIGH}={L, M, H}
2. ZN = {LOW, MEDIUM, HIGH, VERY HIGH} = {L, M, H, VH}
3. INDUS = {VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH}
= {VL, L, M, H, VH}
4. NOX = {VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH }
={VL, L, M, H, VH}
5. RM = {SMALL, MEDIUM, LARGE, VERY LARGE}={S, M, L, VL}
6. AGE = {OLD, MEDIUM, NEW}={O, M, N}
7. DIS = {NEAR, FAR}={N, F}
8. RAD = {SMALL, MEDIUM, LARGE}={S, M, L}
9. TAX = {VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH}
={VL, L, M, H, VH}
10.PTRATIO={ VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH}
={VL, L, M, H, VH}
11.B = {LOW, MEDIUM, HIGH}={L, M, H}
12. LSTAT ={LOW, MEDIUM, HIGH, VERY HIGH} ={L, M, H, VH}

Given all these structural components in place, we complete the gradient-based learning. For illustration, the resulting network for the LOW price is summarized in Table 7.1. Because of the logic transparency of the networks, the meaning of logic description of the data is quite straightforward: low MEDV has a strong association with average room number (RM), pupil-teacher ratio (PTRATIO), population (LSTAT) and accessibility to radial highways (RAD). Some other inputs such as per capita crime rate (CRIM), built year of houses (AGE) are less essential. For instance, in Rule 3, we can note that the higher LSTAT implies lower price. Also, we can see that the proportion of blacks (B) is always weighted quite high, implying that it has no effect to the low MEDV. The rest of the expression can be interpreted in a similar manner.

Table 7.1 The interpretation of the network for MEDV = L

Context: MEDV = LOW	
Performance index (RMSE)	Train = 0.1862 Test = 0.2442
Rules	[(RM is M)$_{0.00}$ and (RAD is M)$_{0.00}$ and (AGE is N)$_{0.28}$ and (TAX is M)$_{0.33}$ and (NOX is M)$_{0.50}$ and (LSTAT is H)$_{0.71}$] $_{1.00}$ OR [(PTRATIO is H)$_{0.00}$ and (CRIM is M)$_{0.24}$ and (LSTAT is H)$_{0.58}$ and (AGE is N)$_{0.85}$ and (DIS is N)$_{0.92}$ and (NOX is H)$_{0.98}$] $_{1.00}$ OR [(LSTAT is H)$_{0.00}$ and (AGE is N)$_{0.34}$ and (INDUS is H)$_{0.38}$ and (ZN is L)$_{0.65}$ and (NOX is H)$_{0.90}$] $_{1.00}$

Possible tradeoffs between accuracy and compactness of the logic description is achieved by analyzing the values of the performance index while reducing the model and retaining a certain number of rules (K) and keeping some limited number of the conditions (L). The results shown in Figure 7.5 indicate that there are some values of these parameters at which the performance index does not increase while the structure has been reduced.

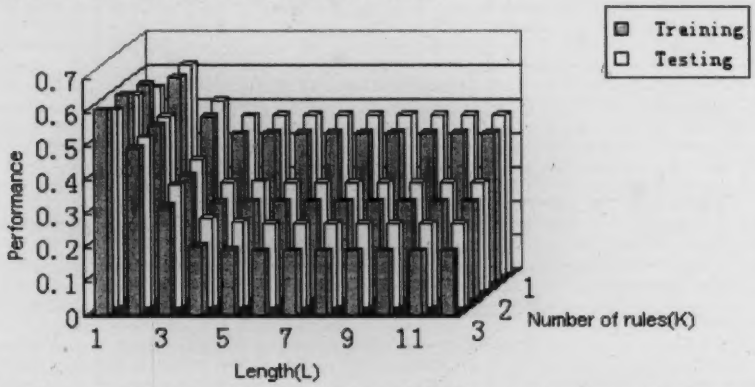


Figure 7.5 Values of the performance (training and testing set) treated as a function of K and L

By inspecting the changes in the values of the performance index choose the most three important rules and no more than 4 conditions to interpret the network. This produces the following description of the data

Context:
MEDV = LOW
Rules:
[(RM is M)$_{0.00}$ and (RAD is M)$_{0.00}$ and (AGE is N)$_{0.28}$ and (TAX is M)$_{0.33}$] $_{1.00}$
OR
[(PTRATIO is H)$_{0.00}$ and (CRIM is M)$_{0.24}$ and (LSTAT is H)$_{0.58}$ and (AGE is N)$_{0.85}$] $_{1.00}$
OR
[(LSTAT is H)$_{0.00}$ and (AGE is N)$_{0.34}$ and (INDUS is H)$_{0.38}$ and (ZN is L)$_{0.65}$] $_{1.00}$

In a similar way, we interpret the model for the two other contexts. By considering the accuracy and compactness of the logic expressions, we choose the essential subsets of conditions and rules. Table 7.2 summarizes two the most important rules for each context with at most four conditions in each rule.

Table 7.2 The interpretations of networks for MEDV = M and H

Context: MEDV = MEDIUM

Performance index (RMSE)	Train =0.2843 Test = 0.3041
Rules	[(RM is M)$_{0.00}$ and (AGE is N)$_{0.00}$ and (PTRATIO is M)$_{0.02}$ and (CRIM is L)$_{0.20}$] $_{1.00}$ OR [(LSTAT is M)$_{0.00}$ and (RM is M)$_{0.20}$ and (PTRATIO is H)$_{0.22}$ and (AGE is N)$_{0.59}$] $_{0.79}$

Context: MEDV = HIGH	
Performance index(RMSE)	Train =0.3014 Test = 0.3058
Rule	[(RM is L)$_{0.00}$ and (CRIM is L)$_{0.21}$ and (RAD is M)$_{0.79}$ and (ZN is M)$_{0.85}$]$_{1.00}$ OR [(CRIM is L)$_{0.00}$ and (NOX is L)$_{0.22}$ and (RM is L)$_{0.59}$ and (PTRATIO is L)$_{0.64}$]$_{1.00}$

From Table 7.2, we note that real estate of medium price is characterized by medium average room number (RM), comes with newer houses (AGE), and medium status of the population (LSTAT). As the high MEDV, the low crime rates (CRIM), larger average room number (RM) and nitric oxides concentration (NOX) are also the variables that reflect high prices.

7.1.2 Auto-MPG dataset

This experimental data set comes from the UCI Machine Learning repository and deals with the fuel efficiency expressed in miles per gallon (MPG). It has six input variables such as number of cylinders (CYL), displacement (DIS), horsepower (HP), weight (W), acceleration (ACC), and the model year (MODEL). As before, the fuzzy equalization was completed for three fuzzy sets, MPG ={SMALL, MEDIUM, LARGE}={S, M, L}), see Figure 7.6.

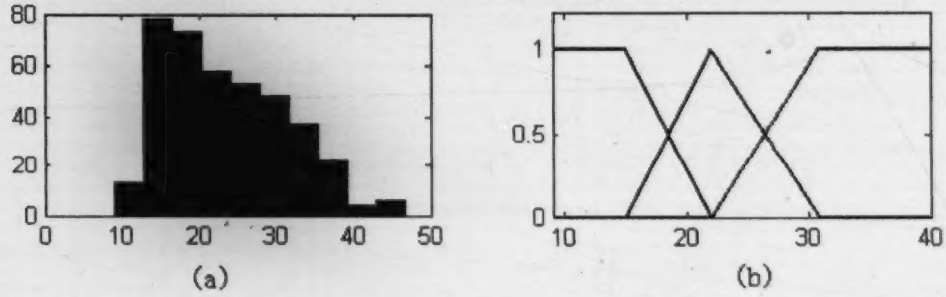


Figure 7.6 Fuzzy equalization of the MPEG output:

(a) histogram, and (b) resulting fuzzy sets

Applying the context-based clustering (with 3 clusters per context), projection and reduction (ε=0.9), we end up with the following linguistic terms formed for each input variable

1. CYL = {SMALL, MEDIUM, LARGE, VERY LARGE} ={S, M, L, VL}
2. DIS = {VERY SMALL, SMALL, MEDIUM, LARGE, VERY LARGE}
   = {VS, S, M, L, VL}
3. HP = {VERY SMALL, SMALL, MEDIUM, LARGE, VERY LARGE}
   = {VS, S, M, L, VL}
4. W = {VERY LIGHT, LIGHT, MEDIUM, HEAVY, VERY HEAVY,
   EXTREMELY HEAVY} = {VL, L, M, H, VH, EH}
5. ACC = {SMALL, MEDIUM, LARGE} = {S, M, L}
6. MODEL = {VERY OLD, OLD, MEDIUM, NEW, VERY NEW}
   ={VO, O, M, N, VN}

The constructed networks come with the interpretation; refer to Table 7.3.

Table 7.3 Interpretation of the network for MPG ={S, M, L}

Context: MPG = S	
Performance index (RMSE)	Train =0.1579 Test = 0.1669
Rule	[(HP is L)$_{0.00}$ and (CYL is VL)$_{0.68}$] $_{1.00}$ OR [(CYL is VL)$_{0.00}$ and (MODEL is O)$_{0.74}$ and (HP is M)$_{0.88}$ and (W is H)$_{0.89}$] $_{1.00}$ OR [(CYL is L)$_{0.09}$ and (MODEL is O)$_{0.51}$ and (DIS is S)$_{0.52}$ and (W is M)$_{0.86}$] $_{1.00}$

Context: MPG = M	
Performance index(RMSE)	Train = 0.2098 Test = 0.2356
Rule	[(W is M)$_{0.21}$ and (MODEL is M)$_{0.35}$ and (HP is S)$_{0.60}$ and (ACC is M)$_{0.77}$] $_{0.71}$ OR [(HP is S)$_{0.00}$ and (ACC is M)$_{0.00}$ and (W is L)$_{0.01}$ and (MODEL is M)$_{0.89}$] $_{0.70}$ OR [(DIS is VS)$_{0.00}$ and (MODEL is O)$_{0.00}$ and (HP is S)$_{0.27}$ and (CYL is M)$_{0.70}$] $_{0.59}$

Context: MPG = L	
Performance index (RMSE)	Train = 0.2158 Test = 0.2226
Rule	$[(\text{DIS is VS})_{0.00}$ and $(\text{MODEL is N})_{0.06}$ and $(\text{CYL is M})_{0.80}]_{0.92}$ OR $[(\text{CYL is M})_{0.00}$ and $(\text{DIS is VS})_{0.00}$ and $(\text{W is VL})_{0.18}$ and $(\text{MODEL is M})_{0.69}$ and $(\text{ACC is M})_{0.92}]_{0.92}$

From Table 7.3, we note that in general vehicles with the larger number of cylinders (CYL), older models (MODEL) and higher horsepower (HP) come with lower fuel efficiency. Likewise, cars with smaller displacement (DIS) with newer built models (MODEL) are characterized by higher fuel efficiency. Medium acceleration (ACC) is strongly linked with medium fuel consumption.

7.1.3 Computer dataset

This data set deals with relative CPU performance, described in terms of the following attributes

1. MYCT: machine cycle time in nanoseconds
2. MMIN: minimum main memory in kilobytes
3. MMAX: maximum main memory in kilobytes
4. CACHE: cache memory in kilobytes
5. CHMIN: minimum channels in units
6. CHMAX: maximum channels in units (integer)
7. PRP: published relative performance (integer)

The performance of the CPU is quantified in terms of three contexts, PRP = {LOW, MEDIUM, HIGH} = {L, M, H}. Their design follows the standard scheme used in the previous examples, see Figure 7.7.

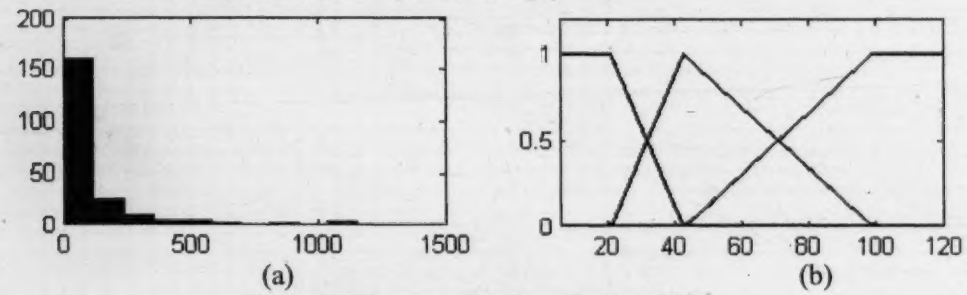


Figure 7.7 Fuzzy equalization of output PRP
(a) A histogram of PRP; (b) Three linguistic labels

Carrying out the context-based clustering (with 3 clusters per cluster), projection and reduction (here ε=0.3), the list of the linguistic terms looks as follows

1. MYCT = {VERY LOW, LOW, MEDIUM, HIGH} = {VL, L, M, H}
2. MMIN = {VERY LOW, LOW, MEDIUM, LARGE, VERY LARGE}
   ={VLW, LW, M, LG, VLG}
3. MMAX = {EXTREMELY LOW, VERY LOW, LOW, MEDIUM, LARGE, VERY LARGE, EXTREMELY LARGE} = {ELW, VLW, LW, M, LG, VLG, ELG}
4. CACHE = {VERY LOW, LOW, MEDIUM, LARGE, VERY LARGE}
   ={VLW, LW, M, LG, VLG}
5. CHMIN = {VERY SMALL, SMALL, MEDIUM, LARGE, VERY LARGE}
   ={VS, S, M, L, VL}
6. CHMAX = {VERY SMALL, SMALL, MEDIUM, LARGE}
   ={VS, S, M, L}

The rule-based description of the data is included in Table 7.4.

Table 7.4 The interpretations of networks for PRP = {LOW, MEDIUM, HIGH}

Context: PRP = LOW	
Performance index	Train = 0.2169 Test = 0.2402
Rule	[(MMAX is ELW)$_{0.00}$ and (CACHE is VLW)$_{0.00}$ and (CHMAX is VS)$_{0.00}$ and (MMIN is VLW)$_{0.07}$ and (CHMIN is VS)$_{0.31}$]$_{1.00}$ OR [(MMAX is LW)$_{0.00}$ and (CACHE is VLW)$_{0.00}$ and (CHMAX is VS)$_{0.00}$ and (MYCT is M)$_{0.24}$]$_{1.00}$ OR [(CACHE is VLW)$_{0.01}$ and (MMAX is VLW)$_{0.04}$ and (CHMAX is VS)$_{0.56}$ and (CHMIN is VS)$_{0.88}$]$_{0.56}$

Context: PRP = MEDIUM	
Performance index	Train = 0.2735 Test = 0.2818
Rule	[(MMAX is LG)$_{0.00}$ and (CACHE is VLW)$_{0.00}$ and (CHMIN is VS)$_{0.10}$ and (MYCT is VL)$_{0.52}$ and (MMIN is VLW)$_{0.98}$]$_{0.65}$ OR [(MMIN is VLM)$_{0.00}$ and (MYCT is L)$_{0.34}$ and (MMAX is VLW)$_{0.82}$]$_{0.61}$ OR [(MYCT is VL)$_{0.00}$ and (CHMAX is VS)$_{0.00}$ and (MMAX is LW)$_{0.15}$ and (CACHE is VLW)$_{0.62}$]$_{0.51}$

Context: PRP = HIGH		
Performance index	Train = 0.2726	Test = 0.2898
Rule	[(MMAX is ELG)$_{0.00}$ and (MYCT is VL)$_{0.22}$]$_{1.00}$ OR [(MYCT is VL)$_{0.08}$ and (CHMIN is S)$_{0.52}$ and (MMIN is LW)$_{0.58}$ and (CACHE is LW)$_{0.96}$]$_{1.00}$	

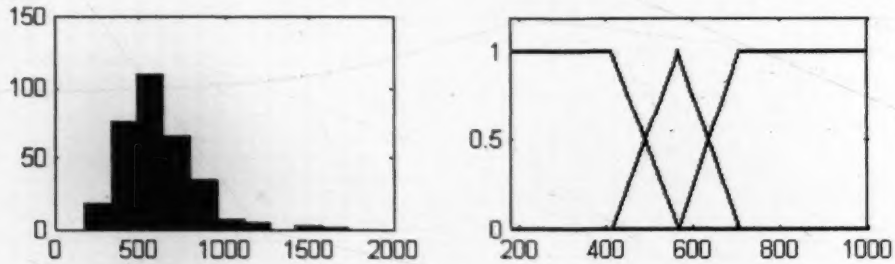
Low maximum main memory (MMAX), low cache memory (CACHE) and small number of maximum channels (CHMAX) imply low CPU performance. With the increase of MMAX and small machine cycle time (MYCT), the performance PRP is enhanced.

7.1.4 Plasma Retinol Levels

This data set comes from [1] and concerns a level of plasma retinol (expressed in ng/ml) whose level varies depending upon a number of factors including age, sex and alcohol consumption, etc. The dataset consists of 315 data and 12 input variables

AGE: Age (years)
SEX: Sex (1=Male, 2=Female).
SMOKSTAT: Smoking status (1=Never, 2=Former, 3=Current Smoker)
QUETELET: Quetelet (weight/(height^2))
VITUSE: Vitamin Use (1=Yes, fairly often, 2=Yes, not often, 3=No)
CALORIES: Number of calories consumed per day.
FAT: Grams of fat consumed per day.
FIBER: Grams of fiber consumed per day.
ALCOHOL: Number of alcoholic drinks consumed per week.
CHOLESTEROL: Cholesterol consumed (mg per day).
RETDIET: Dietary retinol consumed (mcg per day)

We defined three linguistic labels of LOW, MEDIUM, and HIGH, see Figure 7.8.



(a) (b)

Figure 7.8 Fuzzy equalization of RETPLASMA level
(a) histogram and (b) resulting fuzzy sets

For 2 clusters per context and the value of ε equal to 0.9, we end up with the terms

1. AGE ={YOUNG, MEDIUM, OLD} ={Y, M, O}
2. SEX ={MALE, FEMAIL}={M, F}
3. SMOKSTAT ={NEVER, FORMER, CURRENT}={N, F, C}
4. QUETELET ={LOW, MEDIUM, HIGH}={L, M, H}
5. VITUSE ={ FAIRLY OFTEN, NOT OFTEN, NO} ={FOFTEN, NOFTEN, NO}
6. CALORIES ={LOW, HIGH} = {L, H}
7. FAT={LOW, MEDIUM, HIGH} = {L, M, H}
8. FIBER={LOW, MEDIUM, HIGH} = {L, M, H}
9. ALCOHOL={LOW, HIGH} = {L, H}
10. CHOLESTEROL={LOW, MEDIUM, HIGH} = {L, M, H}
11. RETDIET ={LOW, HIGH} = {L, H}

Note that we have reported only the most important rule with no more than 3 conditions per rule with 2 clusters for each context.

Table 7.5 The interpretations of networks for RETPLASMA ={LOW, MEDIUM, HIGH}

Context: RETPLASMA = LOW	
Performance index(RMSE)	Train =0.3951 Test = 0. 4075
Rule	[(RETDIET is L)$_{0.11}$ and (SEX is F)$_{0.47}$ and (CHOLESTEROL is M)$_{0.55}$ and (FIBER is M)$_{0.76}$ and (VITUSE is NOFTEN)$_{0.98}$]$_{0.40}$

Context: RETPLASMA = MEDIUM	
Performance index(RMSE)	Train = 0.3374 Test = 0.3527
Rule	[(CALORIES is L)$_{0.00}$ and (ALCOHOL is L)$_{0.00}$ and (SEX is F)$_{0.57}$]$_{0.32}$

Context: RETPLASMA = HIGH	
Performance index(RMSE)	Train = 0. 4259 Test = 0.4474
Rule	[(CALORIES is L)$_{0.00}$ and (RETDIET is L)$_{0.00}$ and (QUETELET is M)$_{0.21}$ and (FIBER is M)$_{0.57}$ and (FAT is M)$_{0.76}$]$_{0.32}$

By carefully examining the rules in Table 7.5, we come up with a concise

description of the diagnostic nature: female (F) with low dietary retinol consumed (RETDIET) demonstrates low level of plasma retinol. Number of calories consumed per day (CALORIES) and alcoholic drinks consumed per week (ALCOHOL) also have a significant impact on plasma retinol. Medium consumption of fat (FAT) and fiber (FIBER) increase the plasma retinol level.

7.1.5 Air pollution at a road -- NO2 dataset

This dataset, originally collected by the Norwegian Public Roads Administration [2], deals with air pollution at a road. It shows the relationships of traffic volume and meteorological variables. It has 500 observations, and 7 input variables:

NCAR: the logarithm of the number of cars per hour
T1: temperature 2 meters above ground (degree C)
WSPEED: wind speed (meters/second)
T2: the temperature difference between 25 and 2 meters above ground (degree C)
WDIRECT: wind direction (degrees between 0 and 360)
HOUR: hour of day from October 2001 to August 2003
DAY : day number from October 2001 to August 2003

The hourly values of the logarithm of the concentration of NO_2 (particles) are treated as the output of the network. The results are shown in Figure 7.9 and Table 7.6 and 7.7.

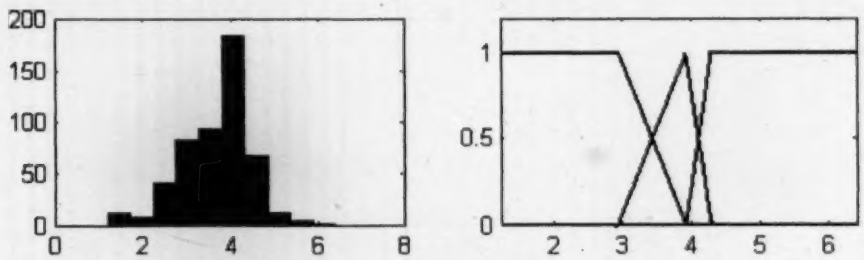


Figure 7.9 Fuzzy equalization of the NO2
(a) the histogram and (b) resulting fuzzy sets

Table 7.6 Fuzzy sets formed in the input space (5 clusters per context; ε=0.7)

NCAR	{EXTREMELY LOW, VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH, EXTREMELY HIGH} = {EL, VL, L, M, H, VH, EH}
T1	{LOW, MEDIUM, HIGH, VERY HIGH } = {L, M, H, VH}

WSPEED	{LOW, MEDIUM, HIGH} = {L, M, H}
T2	{LOW, MEDIUM, HIGH} = {L, M, H}
WDIRECT	{SMALL, MEDIUM, LARGE, VERY LARGE}= {S, M, L, VL}
HOUR	{VERY SMALL, SMALL, MEDIUM, LARGE, VERY LARGE} ={VS, S, M, L, VL}
DAY	{EXTREMELY SMALL, VERY SMALL, SMALL, MEDIUM, LARGE, VERY LARGE} ={ES, VS, S, M, L, VL}

Table 7.7 The interpretation of the networks

Context: NO_2= LOW	
Performance index(RMSE)	Train =0.3439 Test = 0.3614
Rule	[(NCAR is M)$_{0.00}$ and (T1 is H)$_{0.00}$ and (WSPEED is M)$_{0.00}$ and (WDIRECT is L)$_{0.00}$ and (HOUR is S)$_{0.00}$]$_{1.00}$ OR [(WSPEED is M)$_{0.00}$ and (T2 is M)$_{0.00}$ and (WDIRECT is M)$_{0.00}$ and (DAY is L)$_{0.00}$ and (HOUR is S)$_{0.10}$]$_{1.00}$ OR [(T2 is M)$_{0.00}$ and (NCAR is VL)$_{0.19}$ and (WSPEED is M)$_{0.27}$ and (DAY is S)$_{0.78}$ and (WDIRECT is M)$_{0.96}$]$_{1.00}$

Context: NO_2= MEDIUM	
Performance index(RMSE)	Train = 0.3261 Test = 0.3503
Rule	[(T1 is H)$_{0.00}$ and (WSPEED is M)$_{0.00}$ and (T2 is M)$_{0.00}$ and (WDIRECT is M)$_{0.00}$ and (DAY is L)$_{0.00}$]$_{1.00}$ OR [(WSPEED is M)$_{0.00}$ and (T2 is M)$_{0.00}$ and (HOUR is L)$_{0.00}$ and (WDIRECT is L)$_{0.06}$ and (DAY is L)$_{0.82}$]$_{0.72}$ OR [(DAY is VS)$_{0.00}$ and (HOUR is M)$_{0.31}$ and (T1 is M)$_{0.46}$ and (NCAR is H)$_{0.99}$]$_{0.66}$

Context: NO_2= HIGH	
Performance index(RMSE)	Train = 0.3800 Test = 0. 4180

Rule	[(T1 is H)$_{0.00}$ and (WDIRECT is L)$_{0.05}$ and (DAY is L)$_{0.30}$ and (NCAR is VH)$_{0.41}$ and (HOUR is L)$_{0.83}$]$_{0.93}$ OR [(NCAR is VH)$_{0.15}$ and (DAY is M)$_{0.20}$]$_{0.74}$ OR [(WDIRECT is M)$_{0.00}$ and (NCAR is VH)$_{0.25}$ and (T1 is M)$_{0.54}$ and (DAY is VS)$_{0.69}$]$_{0.67}$

This extensive suite of experiments led us to some general observations. The performance of the network on the training and testing set expressed as the ratio of Q_{test}/Q_{train} varies from 101.46% to 131.15% on average the performance on the testing set deteriorated by 30.86%. The networks led to the fairly consistent logic description of the experimental data resulting in 2.33 rules (on average) with an average length of 4.06 variables. Interestingly, we noted that each model used only a portion of all inputs (and this amounts to 27.27%~83.33% of all inputs). This is quite indicative of the redundancy existing in the data where outputs could be quite well described by a limited portion of the inputs.

7.2 Networks constructed by AND and OR unineurons

Our experiments are guided by the three different strategies discussed in Chapter 6. Throughout the experiments, we used 10-fold cross-validation: 90% of the data were used for the training and the remaining 10% were used for the testing; cycles of learning/testing were repeated 10 times. In all experiments, the size of the population of the PSO parameters was equal to 200. The value of the inertial weight (ξ) was set to 0.6. Internal performances are evaluated during the network learning, and external performances are reported for the induced fuzzy data. The experimented datasets in this section are the same as the ones in section 7.1, unless specified otherwise.

7.2.1 Boston Housing dataset

Following the overall development scheme introduced in this study, we start with k-means discretization completed in the output space and PSO discretization of input variables. The discretization results are reported in Table 7.8 where we show the relationship between input and output discretization levels "p" and "c," respectively, and the resulting mean inconsistency rates $\zeta \pm$ a standard deviation.

Table 7.8 Inconsistency rate ζ as a function of p and c.

p \ c	2	3	4	5	6
2	0.04% ± 0.001	0.00% ± 0.001	0.00% ± 0.000	0.00% ± 0.000	0.00% ± 0.000
3	0.66% ± 0.005	0.00% ± 0.000	0.00% ± 0.000	0.00% ± 0.000	0.00% ± 0.000
4	1.18% ± 0.021	0.32% ± 0.007	0.00% ± 0.000	0.00% ± 0.000	0.00% ± 0.000
5	3.07% ± 0.042	0.41% ± 0.011	0.00% ± 0.002	0.00% ± 0.000	0.00% ± 0.000
6	3.81% ± 0.047	0.49% ± 0.009	0.00% ± 0.000	0.00% ± 0.000	0.00% ± 0.001
7	4.74% ± 0.050	2.07% ± 0.012	0.00% ± 0.000	0.00% ± 0.000	0.00% ± 0.000
8	5.09% ± 0.038	1.04% ± 0.017	0.00% ± 0.001	0.00% ± 0.000	0.00% ± 0.000
9	7.34% ± 0.049	1.95% ± 0.020	0.00% ± 0.001	0.00% ± 0.000	0.00% ± 0.000
10	7.78% ± 0.057	1.38% ± 0.013	0.00% ± 0.002	0.00% ± 0.001	0.00% ± 0.000

To examine the quality of the obtained discretization, we compare the above results to the results obtained by means of the simplest discretization, equal-width discretization, which divides the input variable into k intervals with equal size, where k is a predefined parameter. The selected entry for comparison, p = 3 and c = 3, has a minimum value of p+c and zero standard deviation. We apply the equal-width discretization and obtain the inconsistency rates listed in Table 7.9.

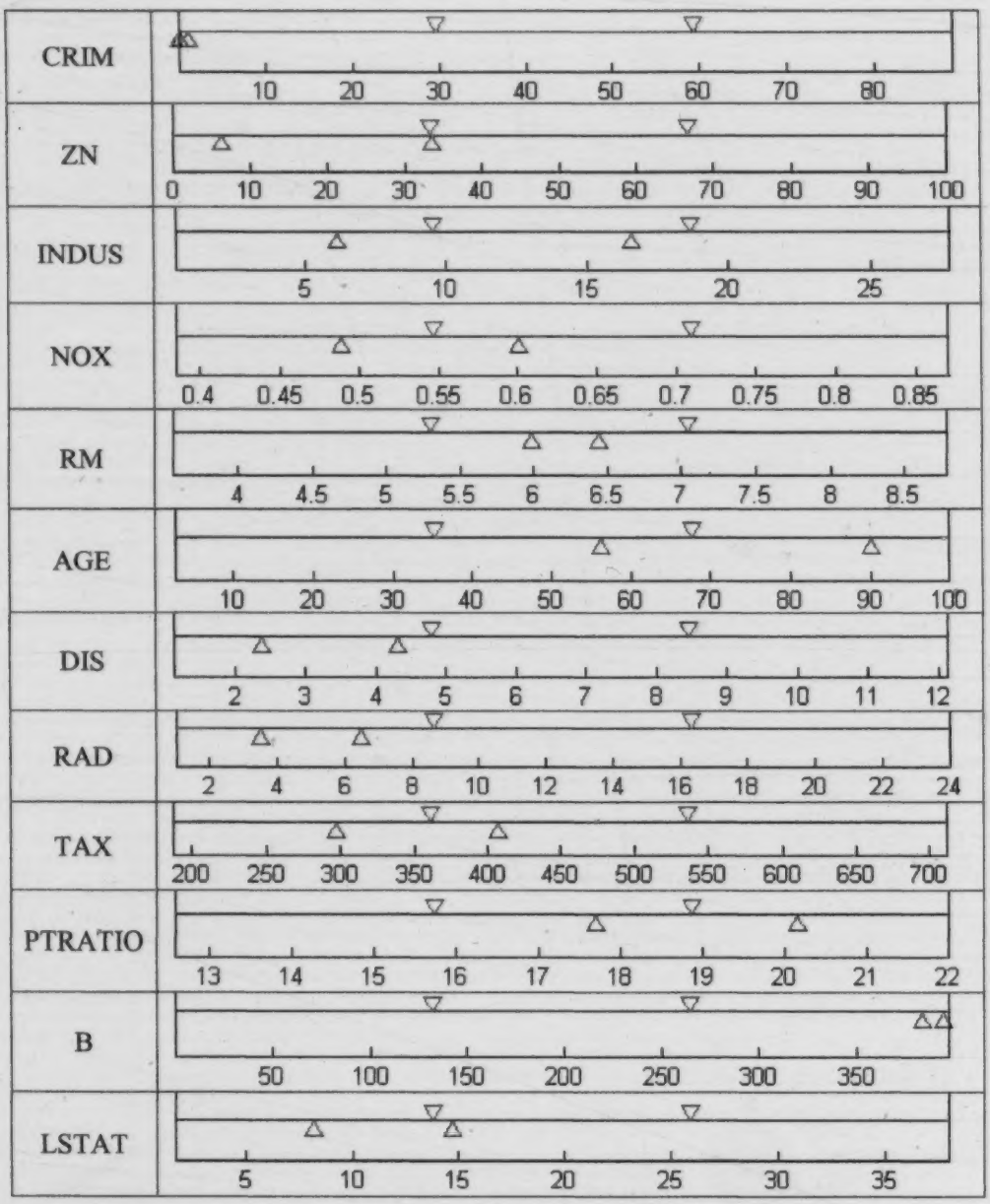
Table 7.9 Comparison of inconsistency rates between two discretization methods.

Discretization method	Training	Testing
PSO + K-Means	0.00% ± 0.000	0.00% ± 0.000
Equal-width discretization	12.02% ± 0.007	3.8% ± 0.022

To illustrate the distribution of cutoff points by these two methods, we randomly selected one training data point and plotted some of the variables along with the cutoff points. Table 7.10 shows the difference between these two methods. PSO cutoff points are denoted with $\triangle$ (triangle up), and cutoff points from the equal-width discretization method are denoted with ∇ (triangle down). Note that the cutoff points for features CRIM and B are grouped very closely due to the distribution of data points of these two features.

Table 7.10 Distribution of cutoff points by PSO $\triangle$ (triangle-up) vs. Equal-width discretization ∇ (triangle-down).

Attribute	Distribution of cutoff points



Having a discrete dataset, we now proceed with the development and training of the network. First we need to determine the number of fuzzy neurons h forming the hidden layer of the network. For experimentation, we arbitrarily select two to ten hidden neurons to illustrate the performance of the network with parameter $[0, 1]^p$, by means of learning Strategy-2. We experimented with this number to demonstrate its impact on the convergence of the method. The values of the

performance index Q obtained are shown in Figure 7.10. The optimal performance index in Figure 7.10 occurs with the number of hidden neurons equal to five. Thus, in our experiments we set the number of neurons in the hidden layer equal to five; that is, h = 5 for all networks.

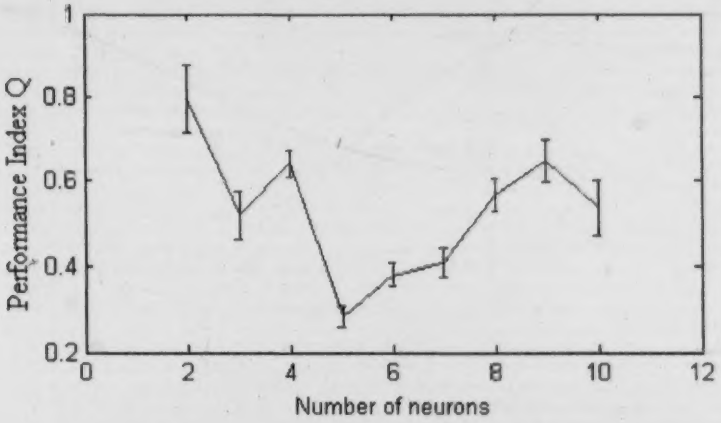
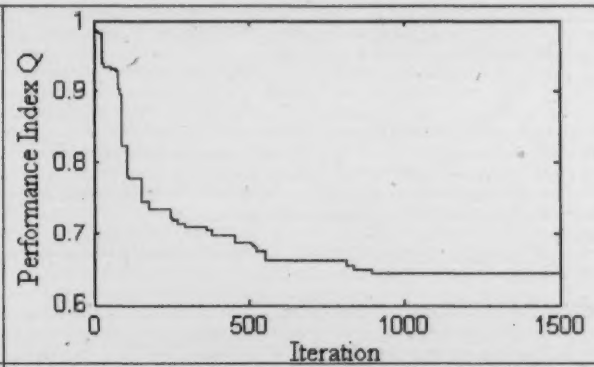


Figure 7.10 The number of neurons in the hidden layer vs. performance index Q

Three learning strategies were then applied to the networks constructed for experimentations. Strategy-1 considers the Boolean parameters, i.e.$\{0,1\}^p$; Strategy-2 and Strategy-3 are both dealing with the continuous parameters that are between 0 and 1, i.e.$[0,1]^p$. Table 7.11-7.13 illustrate the training and testing performance of the network on the housing data using the three learning strategies. The values of networks parameters are reported in Table 7.11 – Table 7.13. Color map for the parameters are also listed in the bottom of the tables. The values of parameters ranges from zero to one, the color map shown in the figure is from dark to light.

Table 7.11 Strategy-1 PSO training Boolean parameters $\{0,1\}^p$

Plot of Training	
Performance Index	0.70 ± 0.06 (Training) 0.74 ± 0.08 (Testing)

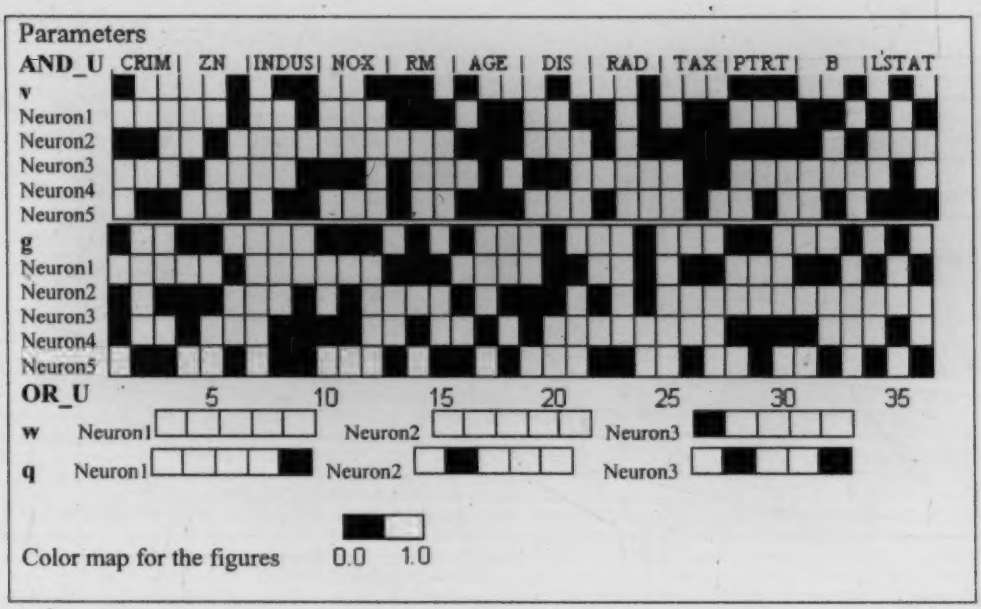
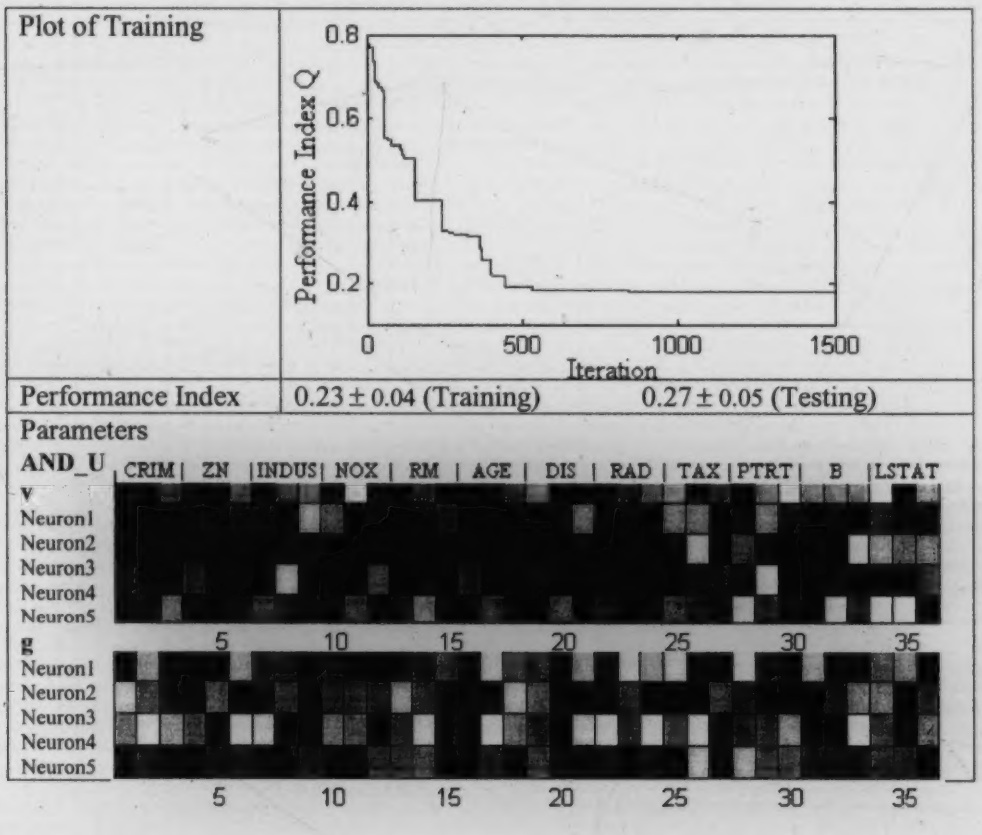


Table 7.12 Strategy-2 PSO training continuous parameters $[0,1]^p$



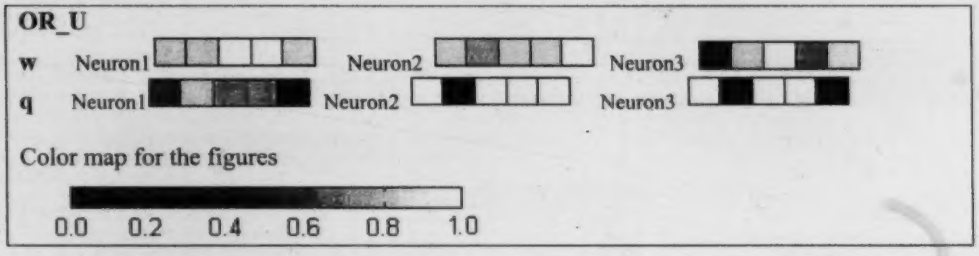
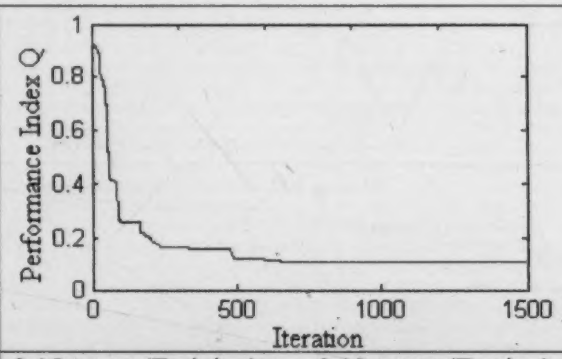
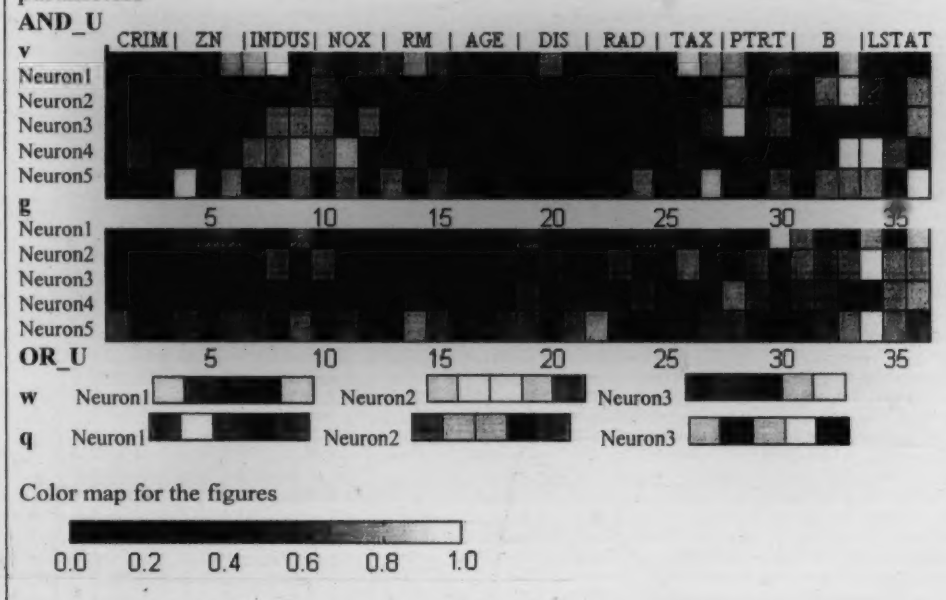


Table 7.13 Strategy-3 Hybrid learning (PSO and gradient-based learning) of continuous parameters $[0,1]^p$

Plot of Training	
Performance Index	0.15 ± 0.04 (Training) 0.19 ± 0.04 (Testing)

parameters



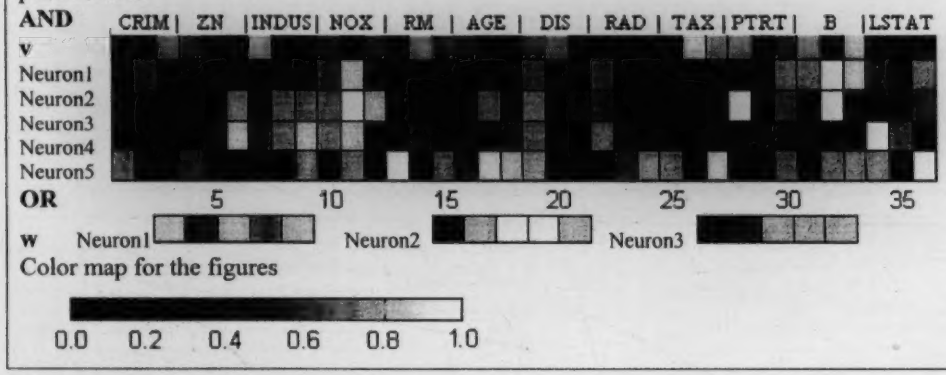
For Strategy-3, we further examine the network interpretability by replacing the AND_U by AND neuron, and OR_U by OR neuron. Only gradient-based learning is applied for parameter optimization. Results for Strategy-3 are compared with results for the AND-OR network.

Table 7.14 AND-OR network trained by gradient-based learning

of continuous parameters $[0,1]^p$

Performance Index	0.17 ± 0.04 (Training)	0.21 ± 0.05 (Testing)

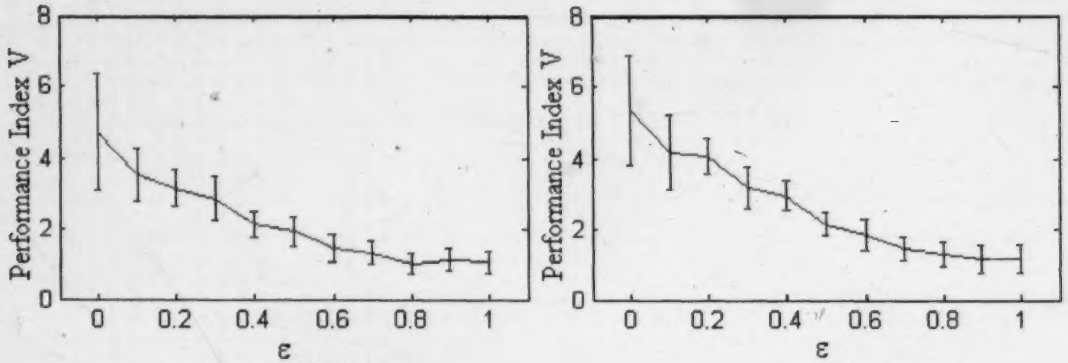
parameters



It is interesting that the network built by AND and OR neurons performs more poorly than the one constructed by unineurons. The following observations can be drawn from the above comparison.

1. The identity points **g** and **q** offer more flexibility to the network so networks built by unineurons perform better that networks constructed with ordinary neurons.
2. The neurons at the output layer, OR and OR_U, act similarly and the weights (w) are all close to 1.
3. For the connections (v) of AND neurons, the lighter color indicates weaker connectivity. Most of the dark areas are located in the blocks where attributes CRIM, RM, AGE, TAX, and PTRATIO are represented, demonstrating stronger connections and identity points for AND_U neurons.

We also examined the fuzzy partition performance index which is expressed as a function of ε (Figure 7.11). Note that ε is the same for all variables in Table 7.11-7.13; all variables use the index shown above and include the result for $\varepsilon = 0$ (which assesses the quality of the Boolean model).



(a) Training (b) Testing

Figure 7.11 Performance index V a function of ε

Figure 7.11 shows the training and testing performance index V for variations of the core parameter ε (ranging from 0 to 1). From Figure 7.11 we find the optimal core parameter ε is approximately 0.8. The lower the value of ε, the worse the performance of the network. Furthermore, these relationships are highly asymmetric and the induced fuzzy model performs much better than the Boolean model. We also examined the performance by means of a triangular membership function for all input variables, the resulting training and testing performance indexes V are 1.32 ± 0.14 and 1.51 ± 0.12, respectively, for the triangular membership function.

7.2.2 Auto-MPG dataset

Similarly, we first discretize this dataset by k-means and PSO. Figure 7.12 shows the inconsistency rate of the different combinations of p and c after the discretization.

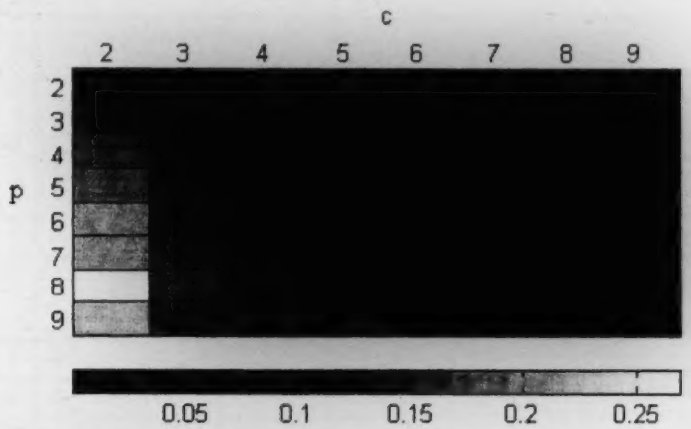


Figure 7.12 Inconsistency rate after discretization

Table 7.15 summarizes the training and testing results of three different learning strategies. Strategy-3 outperforms the other learning strategies in both training and testing.

Table 7.15 Comparison of training and testing performance index Q among three learning strategies

Strategy	Training	Testing
Strategy-1 {0,1}+PSO	0.14 ± 0.07	0.17 ± 0.06
Strategy-2 [0,1]+PSO	0.09 ± 0.01	0.11 ± 0.03
Strategy-3 [0,1]+PSO+gradient-based	0.04 ± 0.01	0.05 ± 0.02
Strategy-3 OR-AND Structure	0.08 ± 0.02	0.11 ± 0.02

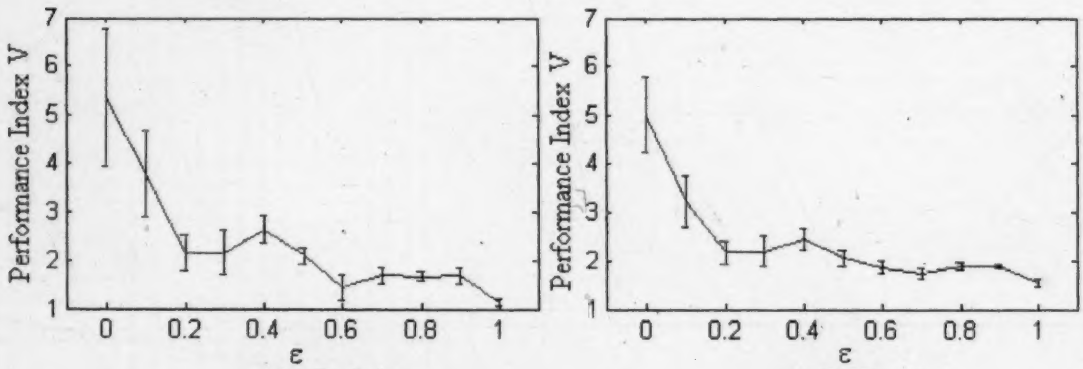


Figure 7.13 Training and testing performance index V vs. core parameter ε.

Figure 7.13 shows the training and testing performance index V for variations of the core parameter ε (ranging from 0 to 1). From this figure we find that the optimal core parameter ε is approximately 1.0. Instead of using the trapezoidal fuzzy membership functions, we use the triangular membership functions for all input variables. The resulting training and testing performance indexes V are 1.73 ± 0.05 and 1.84 ± 0.04, respectively, for the triangular membership functions.

7.2.3 Abalone dataset

The Abalone dataset predicts the age of abalone from the physical measurements shown below.

1. Sex male (M), female (F), and infant (I)
2. Length longest shell measurement
3. Diameter perpendicular to length
4. Height with meat in shell
5. Whole weight grams of whole abalone
6. Shucked weight grams of meat
7. Viscera weight grams of gut weight (after bleeding)
8. Shell weight grams after being dried

The output is given by the number of rings. The number of rings plus 1.5 gives the abalone age in years. Figure 7.14 shows the inconsistency rate of different combinations of p and c after discretization.

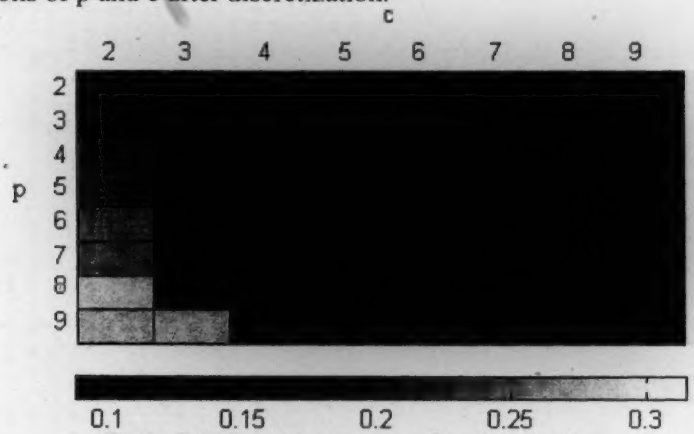


Figure 7.14 Inconsistency rate after discretization

Table 7.16 presents the training and testing results of the three different learning strategies. Strategy-3 gives the best performance index.

Table 7.16 Comparison of training and testing performance index Q among three learning strategies

Strategy	Training	Testing
Strategy-1 {0,1}+PSO	0.17 ± 0.05	0.21 ± 0.06
Strategy-2 [0,1]+PSO	0.05 ± 0.01	0.07 ± 0.02
Strategy-3 [0,1]+PSO+gradient-based	0.04 ± 0.01	0.05 ± 0.02
Strategy-3 OR-AND Structure	0.07 ± 0.02	0.10 ± 0.01

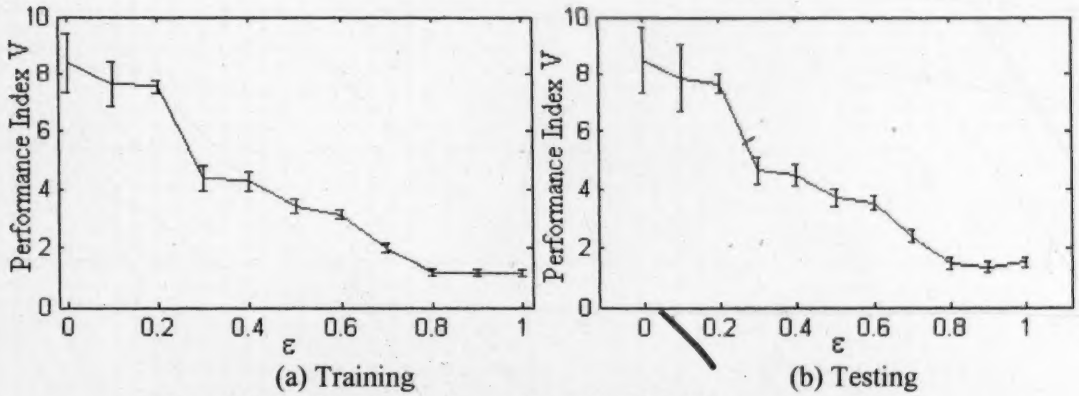


Figure 7.15 Training and testing performance index V vs. core parameter ε.

Figure 7.15 shows the training and testing performance index V for variations of the core parameter ε (ranging from 0 to 1). We find the optimal core parameter ε is approximately 0.8. Instead of using the trapezoidal fuzzy membership functions, we use the triangular membership functions for all input variables. The results of training and testing performance index V is 2.13 ± 0.06 and 2.68 ± 0.05, respectively, for the triangular membership function.

7.2.4 Computer dataset

Figure 7.16 shows the detailed inconsistency rate after discretization.

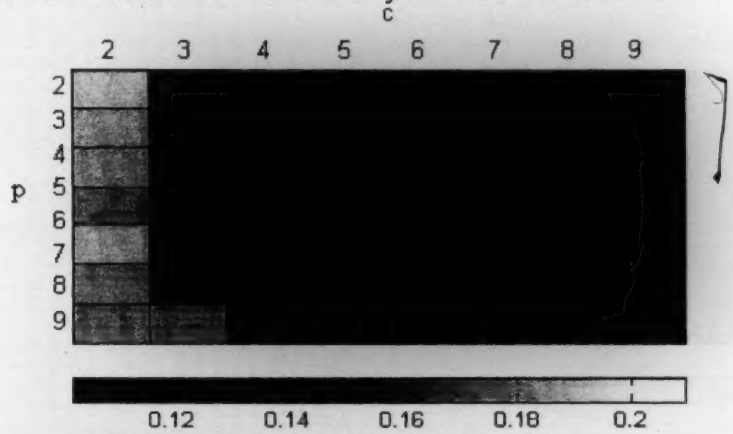


Figure 7.16 Inconsistency rate after discretization

The training and testing results of three different learning strategies are listed in Table 7.17. Again, the performance indexes obtained by Strategy-3 outperform

the other learning strategies in both training and testing.

Table 7.17 Comparison of training and testing performance index Q among three learning strategies

Strategy	Training	Testing
Strategy-1 {0,1}+PSO	0.27 ± 0.05	0.31 ± 0.06
Strategy-2 [0,1]+PSO	0.15 ± 0.02	0.17 ± 0.02
Strategy-3 [0,1]+PSO+gradient-based	0.12 ± 0.01	0.14 ± 0.02
Strategy-3 OR-AND Structure	0.19 ± 0.02	0.21 ± 0.03

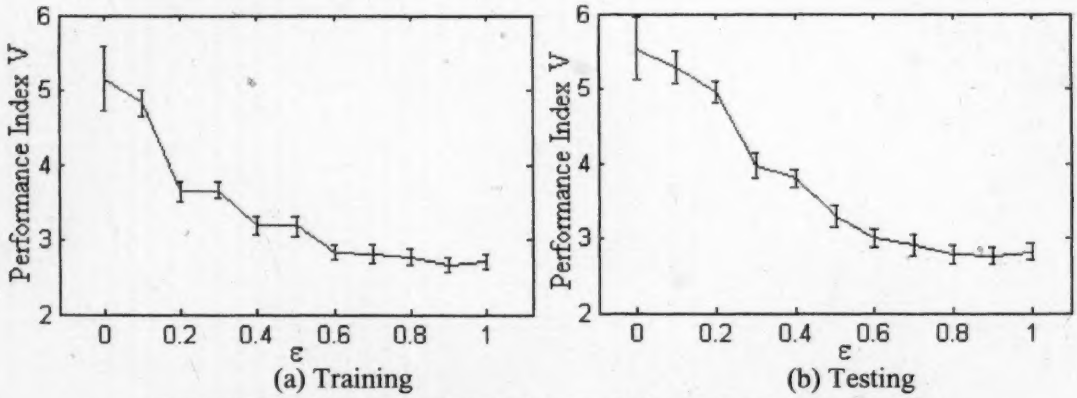


Figure 7.17 Training and testing performance index V vs. core parameter ε.

Figure 7.17 shows the training and testing performance index V for variations of the core parameter ε (ranging from 0 to 1). From Figure 7.17 we find that the optimal core parameter ε is approximately 0.9. Instead of using the trapezoidal fuzzy membership functions, we use the triangular membership functions for all input variables. The training and testing performance index V is 2.7 ± 0.09 and 3.2 ± 0.15, respectively, for the triangular membership functions.

7.3 Conclusions

Extensive experiments have been carried out to express the research methodology and the research schemes adopted in this study. The experimental studies comprise two parts. First, we demonstrate the performance and the generation of logic expressions of the AND-OR network. The user-interactive procedure is included as a simple pruning mechanism to enhance the interpretability of the network while reducing the network size. Second, we discuss the experiments based on the networks built by unineurons. Three

learning strategies are applied for the parametric optimization, and fuzzy partition is introduced to further investigate the network performance.

Bibliography

[1] D.W. Nierenberg, T.A. Stukel, J.A. Baron, B.J. Dain, and E.R. Greenberg, "Determinants of plasma levels of beta-carotene and retinol," American Journal of Epidemiology, vol. 130, pp. 511-521,1989.
[2] http://lib.stat.cmu.edu/datasets/NO2.dat

Chapter 8
Conclusions and Future Work

The tradeoff between accuracy and interpretability is a genuine challenge to the constructs of neurofuzzy computing. Ideally, we would like to see these two modeling requirements being met to the highest extent. The model needs to not only achieve high approximation accuracy for the given data, but it must also realize accurate predictions under unforeseen circumstances. These accomplishments reflect the generalization and predictive capabilities of the developed model. Furthermore the model must be highly transparent so that its users/designers can easily interpret and understand the key relationships captured by it, thus gaining insight and knowledge that was previously unattainable. Given the essential interpretability aspects of the model, a user should be able to interact with it, modifying its structure in order to make further refinements and enhancements.

The ultimate challenge of fuzzy system modeling is to build accurate and transparent models. This thesis presents research in the area and has focused on these fundamental requirements. We showed that the three-component architecture of fuzzy models (where we distinguish between an input interface, a processing core, and an output interface) offers a sound modeling layout using which we could map the modeling requirements. With respect to these three components, various significant research objectives were drawn and completed:

- Investigation of information granulation in the input interface,
- Description of the characteristics of neurons and the resulting neuron-based logic network,
- Exploration of the use of evolutionary techniques in parametric optimization of networks,
- Investigation of the tradeoff between accuracy of the network and interpretability of the logic description,
- Evaluation of the performance of the constructed model by means of the internal and external performance indexes,
- Discussion of dimensionality reduction in the model.

With the outlined objectives in mind, this study has enumerated and carried out various lines of research. In particular, the thesis has delivered:

1. An extensive investigation of existing logic modeling techniques:

A comprehensive literature review was conducted to examine strengths and

weaknesses of existing modeling techniques. Based upon these investigations, we introduced the logic model framework based on fuzzy logic neurons, whose transparency and learning abilities are accentuated to the highest possible extent [11-13]. The resulting network constructs directly benefit from these features which are manifest in the overall network.

2. The development of several techniques in information granulation:

To construct efficient granular information for processing by the fuzzy model, several techniques are employed. First, we use fuzzy equalization and context-based clustering to transform the numeric value to the memberships of its linguistic term. Discretization realized by means of K-Means clustering and particle swarm optimization, is proposed as another data information granulation technique to enhance the transparency of the model.

3. The introduction of merging measurement for the reduction of feature space and further optimization of the granular interface:

Usually, after information granulation, each feature results in a collection of fuzzy sets, and the number of fuzzy sets for each feature is the same. By introducing the merging measurement, we can group the fuzzy sets that satisfy the criterion. By justifying the merging measurement, we end up with a different number of fuzzy sets for each input variable. This in essence reflects the nature of the data and further optimizes the granular interface. In addition, the total number of fuzzy sets is (far) less than the number before merging.

4. The investigation of balance between accuracy and interpretability by means of the pruning mechanism:

The development process of fuzzy networks is highly interactive and user-oriented. By selecting a user defined threshold, we can further reduce the size of the network. Such reduction results in improvement in interpretability of the logic description, and also ensures an acceptable level of accuracy.

5. Comprehensive experimental studies:

The proposed modeling techniques were assessed on real data coming from the UCI machine learning repository. Extensive experimental studies are carried out to demonstrate the feasibility and superiority of the proposed techniques.

For further work we may suggest four main streams of promising research:

1. Exploration of mechanisms of feature (variable) selection:

Feature selection is also known as subset selection or variable selection. Many recent studies in the literature have discussed the application of feature selection methods to high dimensional datasets [1-3,5,7,10,14]. In the presence of high dimensional systems, the challenge of modeling with high accuracy and interpretability are amplified considerably, and the learning of the models requires significantly longer computation times. One possible future direction for our research would introduce feature selection techniques before information granulation. This would reduce feature space through the selection of smaller subsets of interesting features and aid in the interpretation of models while retaining the highest possible degree of accuracy developed on a given dataset. There are two types of feature selection, namely filters and wrappers, as briefly discussed in Chapter 4. Filter methods require less computational effort and thus could constitute a plausible option of preprocessing for information granulation.

2. Exploration of other methods of network learning:

So far we have investigated gradient-based learning and particle swarm optimization. There are other interesting machine learning techniques available which are worth considering. For instance, the memetic algorithm [4,6,8,9] is another population-based approach for heuristic search in optimization problems. Another important issue is the computation time required for training, especially for strategy-3 which is a hybrid algorithm of PSO and gradient-based learning. Perhaps variations in the presented method would result in computationally efficient algorithms. In particular, we could substantially improve learning speed by employing the multi-thread technique or parallel computing during software implementation.

3. Exploration of different structures of the model:

Our research is based on a three-layer logic network. This can be expanded to different constructs, for example, a hierarchical structure, or a combination of fuzzy neurons and unineurons. Changing the topology of the network might improve the model. Further investigation on this issue may involve structure optimization during learning. Consider that a hierarchical structure, the number of levels (the depth of the network), the number of inputs for each level, and the sequence of the input are all structural parameters that need to be optimized before or during parametric learning.

4. Investigation of more sophisticated pruning mechanisms:

Our experiments show the pruning process can improve the interpretability of the network. As a starting point in this research, the pruning mechanism is carried out by setting a threshold for network connections. We could consider more sophisticated design criteria such as a weighted combination of structural

complexity measurement, accuracy measurement, and interpretability measurement, where structural complexity measures complexity of the network structure, the accuracy measurement deals with the correction of input-output mapping, and the interpretability measurement determines the readability of the final rules set. Pruning could also be considered during network learning; this would be more useful if the coupling issue in the model is substantial.

5. Consideration of complexity

There are some limitations in our approach, such as no consideration of the complexity. Usually, granularity is implied by complexity. The choice of a suitable level of granularity will be considered in our future work.

Bibliography

[1] J. Bins, and B.A. Draper, "Feature selection from huge feature sets", 8th IEEE International Conference on Computer Vision, vol. 2, pp. 159-165, 2001.

[2] B. Chakraborty, "Genetic algorithm with fuzzy fitness function for feature selection", 2002 IEEE International Symposium on Industrial Electronics, vol. 1, pp. 315-319, 2002.

[3] K. Kira, and I.A. Rendell, "The feature selection problem: Traditional methods and a new algorithm", 10th National Conference on Artificial Intelligence, MIT Press , pp. 129-134, 1992.

[4] N. Krasnogor, and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy and design issues", IEEE Transactions on Evolutionary Computation, vol. 9, no. 5, pp. 474-488, 2005.

[5] H. Liu, and R. Setiono, "Feature selection via discretization of numeric attributes", IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 4, pp. 642-645, 1997.

[6] P. Merz, "Memetic algorithms for combinatorial optimization problems: Fitness landscape and effective search strategy", Ph. D. Thesis, University of Siegen, 2000.

[7] L.C. Molina, L. Belanche, and A. Nebot, "Feature selection algorithms: a survey and experimental evaluation", 2002 IEEE International Conference on Data Mining, pp. 306-313, 2002.

[8] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms", Caltech Concurrent Computation Program, C3P Report 826, 1989.

[9] P. Moscato, and M.G. Norman, "A 'memetic' approach for the traveling salesman problem. Implementation of a computational ecology for combinatorial optimization on message-passing systems", Parallel Computing and Transputer Applications, edited by M. Valero, E. Onate, M. Jane, J.L. Larriba, and B. Suarez, Ed. IOS Press, Amsterdam, pp. 187-194, 1992.

[10] V. Onnia, M. Tico, and J. Saarinen, "Feature selection method using neural network", 2001 International Conference on Image Processing, vol. 1, pp. 513-516, 2001.
[11] W. Pedrycz, "Heterogeneous fuzzy logic networks: Fundamentals and development studies", IEEE Transactions on Neural Networks, vol. 15, pp. 1466-1481, 2004.
[12] W. Pedrycz, and K. Hirota, "Uninorm-based logic neurons as adaptive and interpretable processing constructs", Soft Computing., vol. 11, no. 1, pp. 41-52, 2007.
[13] W. Pedrycz, "Logic-based fuzzy neurocomputing with unineurons", IEEE Transactions on Fuzzy Systems, vol. 14, no. 6, pp. 860-873, 2006.
[14] D. Zongker, and A. Jain, "Algorithms for feature selection: An evaluation", The 13th International Conference on Pattern Recognition, vol. 2, pp. 18-22, 1996.